Université de Montréal

Identifying Electrons with Deep Learning Methods

par Emre Onur Kahya

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Mémoire présenté à la Faculté des arts et des sciences
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique

Décembre, 2020

Université de Montréal

Département d'informatique et de recherche opérationnelle, Arts et sciences

---

Ce mémoire intitulé

Electron Identification with Deep Learning Methods

Présenté par

Emre Onur Kahya

A été évalué par un jury composé des personnes suivantes

Claude Frasson
Président-rapporteur

Alain Tapp
Directeur de recherche

Jean-François Arguin
Codirecteur

Jian Yang
Membre du jury

Résumé

Cette thèse porte sur les techniques de l'apprentissage machine et leur application à un problème important de la physique des particules expérimentale: l'identification des électrons de signal résultant des collisions proton-proton au Grand collisionneur de hadrons.

Au chapitre 1, nous fournissons des informations sur le Grand collisionneur de hadrons et expliquons pourquoi il a été construit. Nous présentons ensuite plus de détails sur ATLAS, l'un des plus importants détecteurs du Grand collisionneur de hadrons. Ensuite, nous expliquons en quoi consiste la tâche d'identification des électrons ainsi que l'importance de bien la mener à terme. Enfin, nous présentons des informations détaillées sur l'ensemble de données que nous utilisons pour résoudre cette tâche d'identification des électrons.

Au chapitre 2, nous donnons une brève introduction des principes fondamentaux de l'apprentissage machine. Après avoir défini et introduit les différents types de tâche d'apprentissage, nous discutons des diverses façons de représenter les données d'entrée. Ensuite, nous présentons ce qu'il faut apprendre de ces données et comment y parvenir. Enfin, nous examinons les problèmes qui pourraient se présenter en régime de "sur-apprentissage".

Au chapitres 3, nous motivons le choix de l'architecture choisie pour résoudre notre tâche, en particulier pour les sections où des images séquentielles sont utilisées comme entrées. Nous présentons ensuite les résultats de nos expériences et montrons que notre modèle fonctionne beaucoup mieux que les algorithmes présentement utilisés par la collaboration ATLAS. Enfin, nous discutons des futures orientations afin d'améliorer davantage nos résultats.

Au chapitre 4, nous abordons les deux concepts que sont la généralisation hors distribution et la planéité de la surface associée à la fonction de coût. Nous prétendons que les algorithmes qui font converger la fonction coût vers minimum couvrant une région large et plate sont également ceux qui offrent le plus grand potentiel de généralisation pour les tâches hors distribution. Nous présentons les résultats de l'application de ces deux algorithmes à notre ensemble de données et montrons que cela soutient cette affirmation.

Nous terminons avec nos conclusions.

Mots clés: réseaux de neurones, apprentissage automatique, apprentissage de représentations profondes, apprentissage supervisé

Summary

This thesis is about applying the tools of Machine Learning to an important problem of experimental particle physics: identifying signal electrons after proton-proton collisions at the Large Hadron Collider.

In Chapters 1, we provide some information about the Large Hadron Collider and explain why it was built. We give further details about one of the biggest detectors in the Large Hadron Collider, the ATLAS. Then we define what electron identification task is, as well as the importance of solving it. Finally, we give detailed information about our dataset that we use to solve the electron identification task.

In Chapters 2, we give a brief introduction to fundamental principles of machine learning. Starting with the definition and types of different learning tasks, we discuss various ways to represent inputs. Then we present what to learn from the inputs as well as how to do it. And finally, we look at the problems that would arise if we "overdo" learning.

In Chapters 3, we motivate the choice of the architecture to solve our task, especially for the parts that have sequential images as inputs. We then present the results of our experiments and show that our model performs much better than the existing algorithms that the ATLAS collaboration currently uses. Finally, we discuss future directions to further improve our results.

In Chapter 4, we discuss two concepts: out of distribution generalization and flatness of loss surface. We claim that the algorithms, that brings a model into a wide flat minimum of its training loss surface, would generalize better for out of distribution tasks. We give the results of implementing two such algorithms to our dataset and show that it supports our claim.

Finally, we end with our conclusions.

Keywords: neural networks, machine learning, deep learning, supervised learning,

Contents

List of Figures

List of Tables

List of Abbreviations

ATLAS	A Toroidal LHC ApparatuS
BFGS	Broyden-Fletcher- Goldfarb-Shanno
CNN	Convolutional Neural Networks
DNN	Deep Neural Network
EL-ID	Electron Identification
EM	Electromagnetic
ERM	Empirical Risk Minimization
FCN	Fully Connected Network
GAN	Generative Adversarial Networks
IID	Independent and Identically Distributed
INIT	Initialization
KL	Kullback–Leibler
LH	Likelihood
LHC	Large Hadron Collider
LSTM	Long Short Term Memory
MLP	Multi Layer Perceptron
OOD	Out Of Distribution
PAC	Probably Approximately Correct
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristic
SGD	Stochastic Gradient Descent
SM	Standard Model
SWA	Stochastic Weight Averaging
VAE	Variational Auto-Encoder
ViT	Vision Transformer

Acknowledgments

First of all, I would like to thank my parents Orbay and Hurmet, my wife Selva, and my daughters Asude and Erva for bringing happiness into my life.

I would like to thank my adivors Alain Tapp and Jean-François Arguin for their constant support, encouragement and guidance during my studies. I would also like to thank Yoshua Bengio for supporting me and this thesis.

I would also like to thank Pierre McKenzie for helping me to see the beauty of Algorithms.

I would like to thank Dominique Godin for helping me throughout this thesis in various aspects. I would like to thank my friends and colleagues Berkin Malkoc, Jose Gallego, Kazuya Mochizuki, Max Schwarzer, Oleksiy Ostapenko, Sergey Panitkin for all of their help and support.

Finally, I would like to acknowledge the financial support from: IVADO, Calcul Quebec and Compute Canada.

1 Electron Identification Task

In this chapter, we will discuss the electron identification (EL-ID) task. After a brief discussion on the context of our work we will give the motivation behind solving EL-ID problem. Finally, we will discuss the technical properties of the dataset that we used.

1.1 The Large Hadron Collider

The Large Hadron Collider (LHC) is the largest machine and the biggest scientific project in the world. It was built between 1998 and 2008, below the surface at Geneva, close to the French-Swiss border. It is inside a tunnel of circumference size of approximately 27 km. It is a high energy particle accelerator that can speed protons up to 3 m/s short of the speed of light. More than ten thousand scientists from more than a hundred countries are affiliated with the LHC.

The motivation behind building the LHC is to study and to understand the fundamental laws of physics. The laws of physics are different at different scales. We can explain the planetary motions almost perfectly with Newton's laws, but at cosmological scales, we need Einstein's equations. Similarly, at human scales, we can describe how particles move when a force act upon them with Newton's force equations. But at the atomic level, we need quantum mechanics to understand the interaction between molecules.

Similarly, the laws of physics depend on the scale of energy as well. If we look at de Broglie's formula $\lambda p = h$ (h is Planck's constant), we can see that distance scale (wavelength) λ and energy/momentum scale p are opposite of each other. Heuristically this would give the following duality: small distances correspond to large energies, and large distances correspond to small energies. Therefore, to understand the fundamental laws of physics at all scales, we need to go to higher

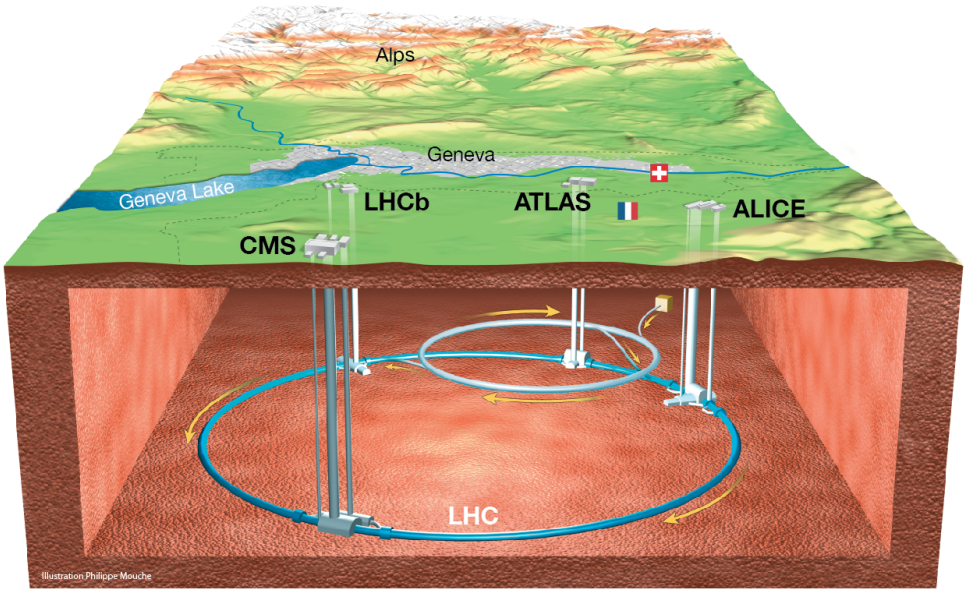

Figure 1.1 – 3D view of Large Hadron Collider and its four major detectors: ALICE, ATLAS, CMS and LHCb. (Image by Mouche, Philippe via CERN OPEN Photos)

energies than what we naturally observe. This was the main motivation behind building particle accelerators.

One way that can reach these energies is by accelerating hydrogen ions with linear accelerators and then stripping the electrons from them, thus producing high energy proton beams. We choose protons because, among all heavy fundamental particles, they are the most stable ones. Therefore when accelerated, they reach velocities extremely close to the speed of light and still not decay to other particles. The other advantage of using protons is that the path of the proton beams can be controlled by huge magnets before billions of them collide every second. And we can repeatedly collide proton beams as they follow opposite circular paths.

For the last hundred years, we were able to understand most of the properties of fundamental particles of physics by analyzing the data coming from particle accelerators. We now have a Standard Model (SM) of particle physics that consistently describes three fundamental forces of nature: electromagnetic, strong and weak interactions. But we still needed to understand some missing parts of the Standard Model, such as the Higgs boson. We also need to look beyond the SM

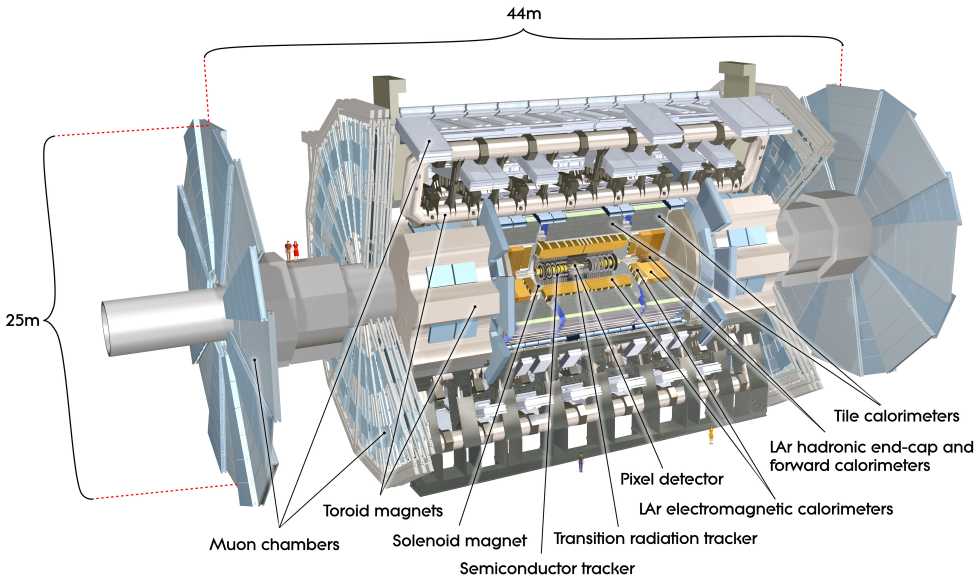

Figure 1.2 – Computer generated image of the whole ATLAS detector(Image by Joao Pequenao via CERN PhotoLab)

of particle physics in order to incorporate gravity into the full picture. Observing and understanding the nature of dark matter particles would be one of the most important achievements of the last century. The LHC, the most powerful particle accelerator, was built for these reasons.

Many scientific findings and discoveries were made with the LHC. But the most famous discovery of the Higgs particle was achieved only three years after it started to operate. In the year 2013, the discovery of the Higgs particle was rewarded with a Nobel Prize in Physics, after ATLAS Collaboration (2012) and CMS Collaboration (2012) published their results.

We came along way, but the search for discoveries still continues at the LHC on various fronts. Precision measurements of various particles, search for supersymmetry and dark matter particles are just a couple of very exciting topics that the LHC could provide answers to.

1.2 ATLAS detector

ATLAS is one of the four major detectors at LHC that is shown in Figure 1.1. It is a general-purpose 48 m long, 25 m wide and 25 m high cylindrical-shaped particle detector. With 7000 tonnes of mass, it is the largest volume particle detector that exists. More than 3000 scientists from most of the major universities and institutes from 38 countries are involved in ATLAS experiments. The University of Montreal ATLAS group (ATLAS-UdeM) is one of the groups that participate in the ATLAS experiments.

Inside the ATLAS detector, there are multiple systems, as we can see from Figure 1.4. Let us briefly discuss these systems starting from one that is closest to the nominal interaction point towards the outermost one. At the innermost part, we have a tracking system whose purpose is measuring the trajectory (including velocity and momentum) of charged particles with them losing as little energy as possible. Then we have an electromagnetic (EM) calorimeter that stops almost all electrons and photons by absorbing their energies. Next level, we have a hadronic calorimeter which stops all the hadronic particles (e.g. protons) that are made out of quarks.

Finally, we have the muon spectrometer at the outermost part of the ATLAS detector. We need a muon spectrometer because muons are the only particles (other than neutrinos and possibly dark matter particles) that can escape from both calorimeters. Muons are electron-like particles, just that they are 200 times more massive than electrons.

The size of the ATLAS data is enormous. ATLAS detector produces one petabyte of raw data at each second. But there is a system, called the trigger, that selects in real-time only the most interesting events, which are then written to disk permanently, that amount to thirty petabytes per year. These huge numbers are anticipated since a billion protons collide at each second. And once they collide, thousands of particles are produced, and they leave tracks on millions of electronic channels [1] that are close to the interaction point.

Just by looking at the size of the data that we have, we can conjecture that machine learning should be very helpful in ATLAS data analysis. Machine learning

1. The Pixel Detector that is shown in Figure 1.2 has 80 million pixels (80 million channels) and has 15 kW power consumption.

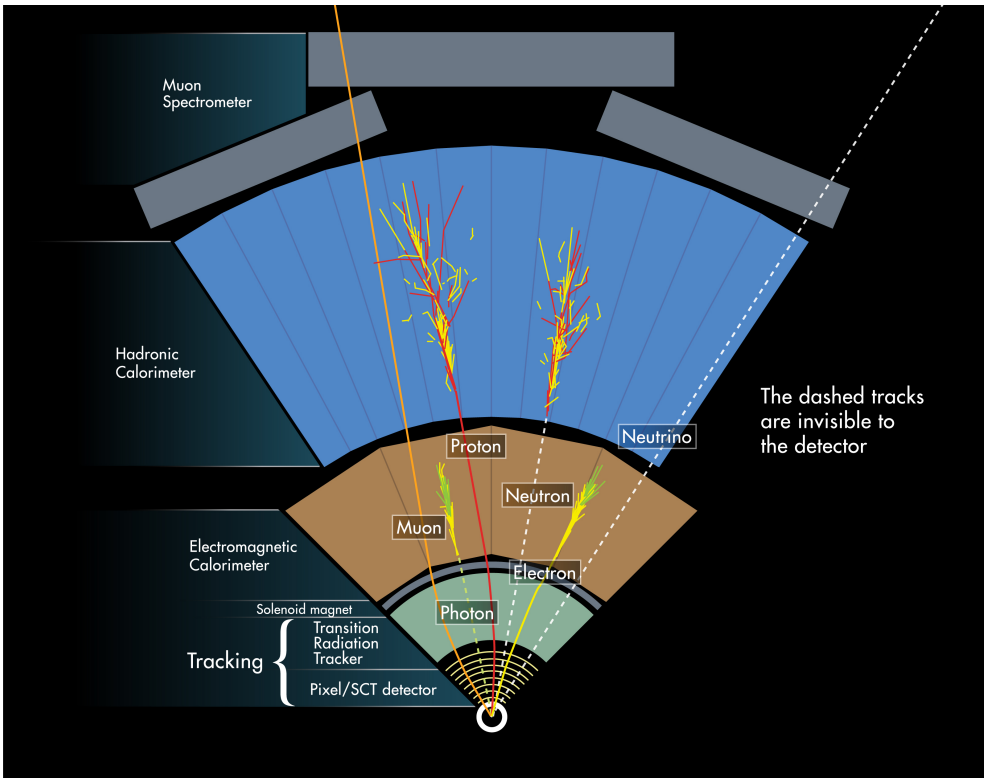

Figure 1.3 – Various particle paths at ATLAS (Image by Joao Pequenao and Paul Schaffner via CERN PhotoLab)

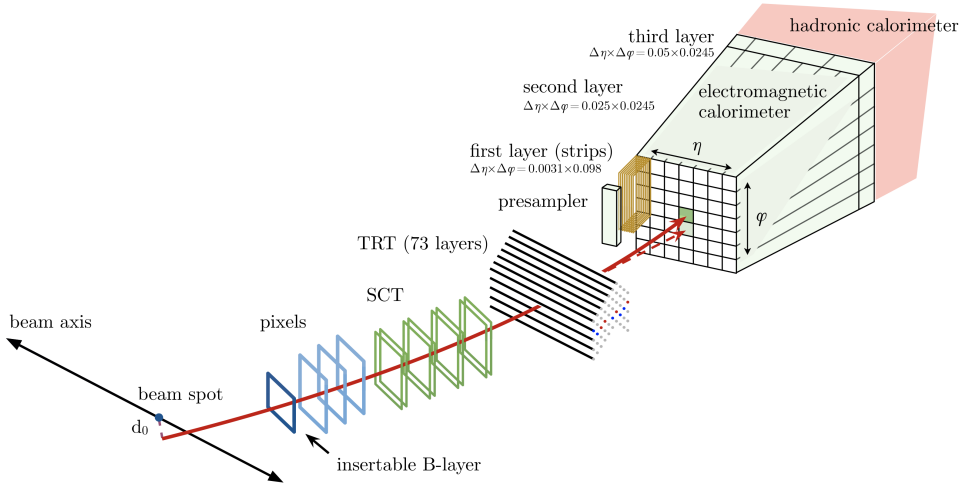

Figure 1.4 – The path of an electron and a photon through ATLAS. The image shows the path that an electron (red line) and a photon (dashed red line) takes starting from the interaction point until it reaches to electromagnetic calorimeter. It also depicts various systems that are inside ATLAS, from the innermost tracking system towards the hadronic calorimeter (Image by ATLAS Collaboration).

algorithms were used early on in high energy physics, starting with the pioneering work of Baldi et al. (2014) and now used more extensively.

1.3 Identifying Electrons

Electrons are one of the most important parts of collider physics studies. We use them to understand the properties of various properties of SM particles more precisely (precision measurements). They also had significant importance for the discovery of the Higgs particle. We are also hopeful to use them as tools for models beyond SM and possible discovery of supersymmetry if any of these particles have electron related signatures. Electrons signatures arise since very interesting particles like top quarks, W and Z bosons, the Higgs boson and putative particles (like from Supersymmetry) are not stable and decay immediately to lighter particles, and often electrons.

There are electrons that come from the collision as well as background electrons. Electrons that come from the collision follow a path that is depicted in Figure 1.4.

Let us describe the electron signature in more detail.

A single track (need to define a track before) geometrically and energetically matched to an energy deposit in the electromagnetic calorimeter. The energy deposit in the calorimeter is relatively narrow, and little to no energy is expected in the electromagnetic calorimeter.

The most abundant background is charged hadrons, like the proton, which are by far the most commonly produced particles at the LHC. They produce a wider energy deposit in the calorimeter and will typically deposit a fraction of their energy in the hadronic calorimeter.

There are also types of backgrounds constituted of real electrons, but that comes from the decay of particles that are typically considered as background at the LHC, like photon conversion or heavy-flavour jets. The signature for these background electrons will be similar to signal electrons except that they will typically contain more than one track and will have wider energy deposits.

In order to analyze the properties of various particles that are produced after the collision, it is very important to distinguish them from the backgrounds of other particle types. In our data-set light-flavour jets, photon conversions, and heavy-flavour jets are the three most abundant backgrounds to electrons.

Background rejection is done by using either a cut-based or a likelihood (LH) based algorithm. They both use tracking and calorimeter variables to distinguish electrons from background particles. Cut-based algorithms apply some cuts based on certain criteria that are peculiar to either electrons or background particles. In both of these algorithms, human-designed variables are used as the input. On the other hand, LH algorithm efficiency is based on these variables to be uncorrelated which is not the case for the electron identification task. Despite these disadvantages, LH is still the currently used electron identification algorithm in ATLAS.

In short, LH is the standard that we want to outperform with the machine learning algorithms that we will discuss in this thesis. We believe that by using lower-level information, such as the tracks and calorimeter images as well as the physical variables, we can get better results with machine learning algorithms.

1.3.1 Likelihood Identification Method

The likelihood estimate is based on the conditional probability of signal or background events given the PDFs for various quantities. These quantities are related to certain parameters or values related to physical conditions that arise during the passage of particles, e.g. number of hits in the innermost pixel layer or the ratio of the energy of the first layer to the total energy in the EM calorimeter.

First, we make the simplification by neglecting the correlation between quantities that we choose for LH estimate. Then the conditional probability of the signal ($L_S(\mathbf{x})$) and the background ($L_B(\mathbf{x})$) are given by the product of the PDFs as

$$L_S(\mathbf{x}) = P(S|\mathbf{x}) = \prod_{i=1}^{n} P(S|x_i) \tag{1.1}$$

$$L_B(\mathbf{x}) = P(B|\mathbf{x}) = \prod_{i=1}^{n} P(B|x_i), \tag{1.2}$$

where $\mathbf{x}$ is a vector where the components are the quantities that we mentioned above. $P(S|x_i)$ is the value of the PDF of the signal given certain quantity that is represented by i at its value x_i. Similarly $P(B|x_i)$ is the background PDF. The PDFs are obtained from binned histograms of each chosen quantity for LH estimate. We then use an adaptive kernel density estimation method to smooth these histograms using TMVA tools (Hoecker et al., 2007).

We use the quantity called likelihood discriminant which is the ratio of signal to total as:

$$d_L = \frac{L_S}{L_S + L_B}. \tag{1.3}$$

To make discriminant behave numerically better especially at places where there is no signal one uses the following quantity

$$d_L^{'} = -\tau^{-1} \ln(d_L^{-1} - 1). \tag{1.4}$$

Here τ is a parameter that is fixed to 15 by ATLAS Collaboration (2014) and $d_L^{'}$ is called the transformed discriminant.

The likelihood method of classification has advantages over cut-based methods. Losing signals at the tails and the impossibility of being able to put cuts

for certain distributions even if their distributions peaks are very different are two disadvantages of cut-based methods. But the assumption that we made about the likelihood, uncorrelated variables is not a good one for electron identification. That is why we would expect to do a much better job with machine learning algorithms.

1.4 Properties of EL-ID Data

The EL-ID task data that we used consists of three different major parts: calorimeter images, track information and scalar quantities. Let us start giving some detail about the image pixels which correspond to the calorimeter channels. As one can see from Figure 1.4, the first image comes from the presampler. After the presampler, there are three images that correspond to EM calorimeter channels. Finally, there are three more images from hadronic calorimeter channels. Out of these seven images, the only high granularity 56x11 pixel-sized image comes from 1^{st} electromagnetic calorimeter (EM) layer. All of the other images' all have pixel sizes of 7x11.

Track information is also used to distinguish electrons from background events. The final part of our data consists of a scalar part that is related to physical quantities related to the collision. These 15 scalar quantities are also used as inputs to the LH based algorithm (ATLAS Collaboration, 2019).

The angular positions of pixels for calorimeters in Figure 1.5 and 1.6 are all given in terms of η and ϕ. These coordinates are chosen due to the cylindrical shape of the ATLAS detector. Here the angle ϕ is the azimuthal angle, and the other angle pseudorapidity η [2] is related to the more familiar polar angle θ

$$\eta \equiv -\ln \tan \frac{\theta}{2}. \tag{1.5}$$

It is worth mentioning the graphical structure of the images. The images that we showed in Figure 1.5 are the random images that we feed into our model. They

2. As a measure of angle particle physicists use pseudorapidity η instead of rapidity. This because it has a simpler form compared to rapidity, but also it is equal to rapidity for high energies. And true rapidity is a measure of an angle that has a peculiar property, where differences of them are invariant with respect to Lorentz boosts along the longitudinal axis.

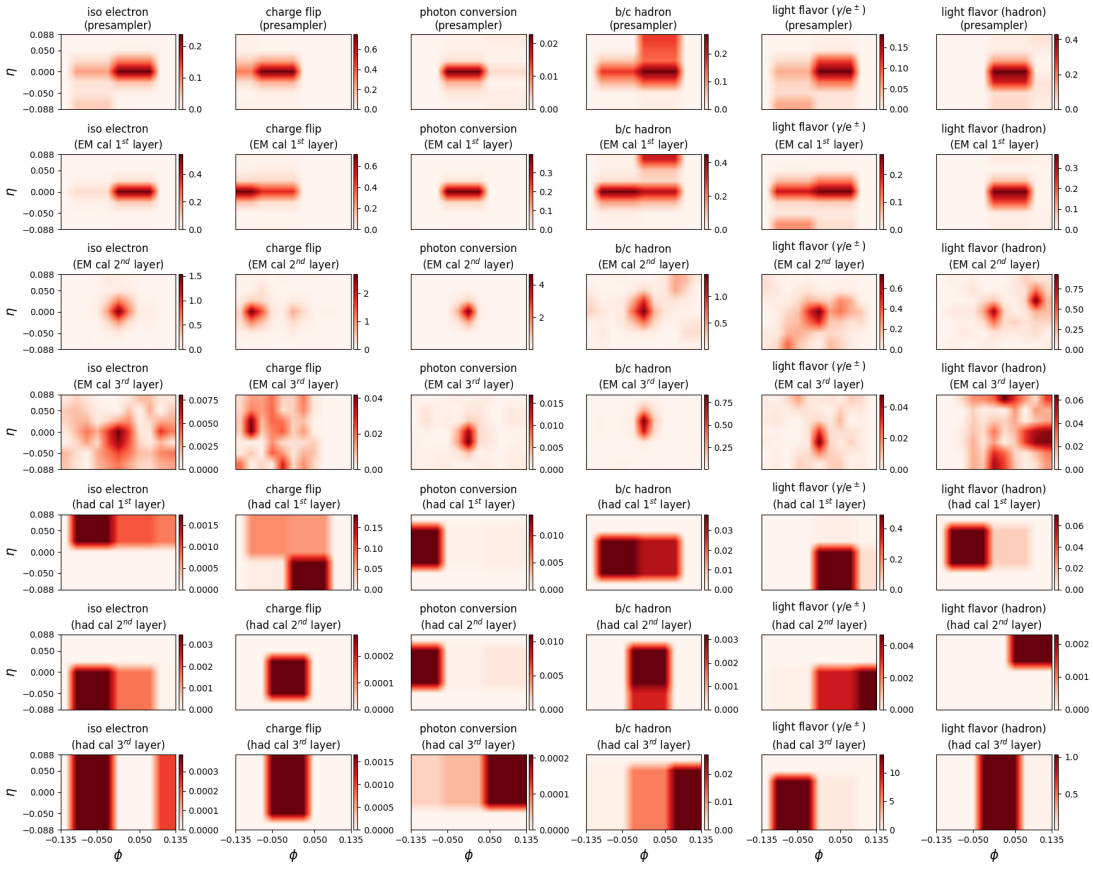

Figure 1.5 – Randomly selected calorimeter images of 6 classes from 7 layers of ATLAS detector (Courtesy of Dominique Godin)

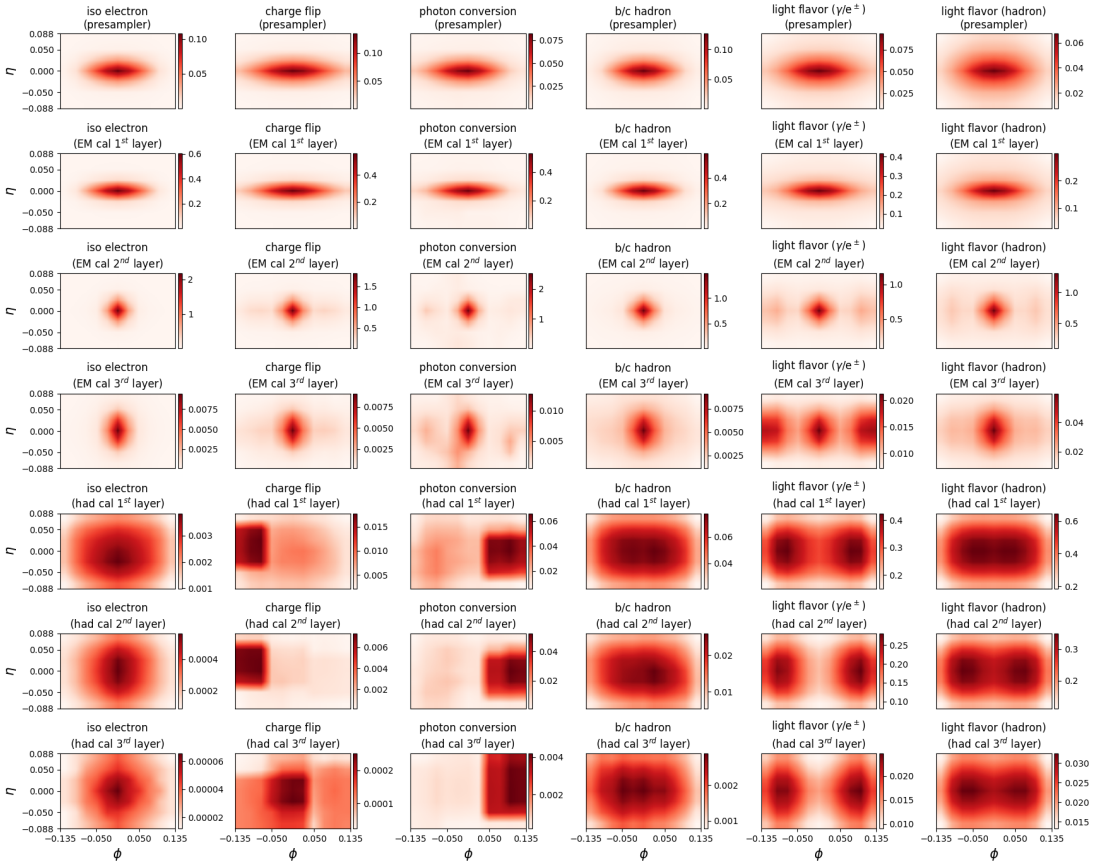

Figure 1.6 – Mean Calorimeter Images (Courtesy of Dominique Godin)

look like randomly distributed coloured patches. But if we look at the mean of these images in our dataset, we can see clear structures appearing in Figure 1.6.

The data samples are provided to our team by the ATLAS Copenhagen group. All of the data come from Monte Carlo based generators that model both the SM physics as well as the detector. The total number of particles that we used is 14675640. There are six classes in our dataset, and the type of background is useful information to optimize the data analysis downstream. Nonetheless, there are six imbalanced classes where two of them make 98.3% of the whole dataset.

In this work, we aim to isolate the signal electrons that make 36.50% of all particles that we have in our dataset. The currently used method for electron identification in the ATLAS detector is the LH algorithm. Hence our goal in this thesis is to achieve high accuracy results in this binary classification task and outperform the currently used LH algorithm.

2 Machine Learning Fundamentals

In this chapter, we will define and review the fundamentals of machine learning. It is not meant to be a complete review of the whole colossal body of work, but rather a short survey of the ideas that are either used or are related to what was done in our thesis work. We refer to Goodfellow et al. (2016) for a more complete review of the fundamentals of machine learning.

2.1 Introduction

Machine learning has been widely and successfully used in many branches of science, ranging from self-driving cars to classification of particles in colliders, from medical diagnosis to identification of astrophysical objects. Machine learning is a collection of algorithms that uses some data to produce some predictive functions to apply those to unseen data or make individual decisions in some environment.

Machine learning uses many tools of mathematics, some of which are: linear algebra, probability, statistics, information theory, game theory, differential geometry. There are also various concepts and tools of general physics (though notably dominated by statistical physics) that are also used either exactly or similarly in machine learning. Some of the commonly used ones are energy, momentum, acceleration, Helmholtz free energy, entropy, spin glasses and many others.

In the next subsection, we will go into some detail to define and explain what learning is. We will do so by discussing the types of various learning tasks.

2.2 Learning Tasks

There are various tasks in machine learning based on the properties of the inputs and our objective. We will discuss each task in some detail.

2.2.1 Supervised Learning

Let us first define an input space that consists of all d-dimensional vectors $\mathbf{x}$ as $\mathcal{X}$. The output space consists of all labels, and we will call it $\mathcal{Y}$. The set of learning examples S consists of n learning examples

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}, \tag{2.1}$$

where each ith example is an element of the direct product of input and output spaces $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. The objective of a supervised learning task is predicting y from input $\mathbf{x}$. This corresponds to finding a function f, which is a map from an input space to an output space

$$f : \mathcal{X} \to \mathcal{Y}. \tag{2.2}$$

There are two main supervised learning tasks, classification and regression. In the classification task, y stands for a class label (either binary or multi-class), and our job is to create a model to predict the class label given a particular input. The target y of the regression task is predicting a real number given a particular input. One can also think of the type of models to achieve these supervised tasks as predictive models.

2.2.2 Unsupervised Learning

Unsupervised learning is the task of modelling the properties of unlabeled data. The models that we construct for unsupervised learning tasks would have descriptive rather than predictive nature. Modelling the distribution of a set of inputs (density estimation) is one type of an unsupervised learning task. Another common unsupervised learning task is discovering the structure of the inputs. Clustering, anomaly detection or dimensionality reduction can all be considered as unsupervised learning tasks.

2.2.3 Semi-supervised Learning

The goal of semi-supervised learning is the same as supervised learning. But these semi-supervised learning tasks are achieved by using both labelled and unlabeled data.

2.2.4 Generative Learning

The goal of discriminative learning algorithms in a supervised learning task is to learn a map from $\mathcal{X}$ to $\mathcal{Y}$. This corresponds to learning $p(y|x)$, from a probabilistic point of view. In the case of generative learning, we would like to model $p(x|y)$ after modelling $p(y)$ that corresponds to class priors. Finally, we use the Bayes rule to apply what we learned to calculate $p(y|x)$ to predict the label y of given an input data x.

2.2.5 Reinforcement Learning

Reinforcement learning tasks consist of learning the collection of optimal actions performed by a so-called agent, where the goal is increasing the cumulative reward. The techniques used in reinforcement learning algorithms are very similar to those of dynamical programming. But this learning algorithm is quite different from other ones since the action that one takes, and the final reward (as winning or losing the game) are very distant.

Let us take an example of binary image classification. In the case of supervised learning, our learning model will either classify particular data correctly or incorrectly. Therefore the "action" that we take is either successful or not. But in the case of reinforcement learning, it is a combination of many sequences of actions, and it is difficult to gauge the success of each action. In a game where one makes 200 moves, was it 50^{th} move or 100^{th} move that made the difference is hard to know.

2.3 How to represent the Inputs

As we discussed in the previous section, our objective ranges from finding maps from the input spaces to the output spaces to estimate the density of the inputs. The first simple way of mapping an input vector $\mathbf{x}$ to an output value is by multiplying it (called dot product) by a co-vector $\mathbf{w}^T$. That would give us some scalar number and we can always add a number without violating linearity condition. Let us express this by explicitly putting the indices of the components as

$$a(\mathbf{x}) = \sum_{i=1}^{d} w_i x_i + b. \tag{2.3}$$

Here the $\mathbf{w}$ term is called the weight since its job is weighing each component of the input vector $\mathbf{x}$ differently. The additive term b is called bias since it biases the function's value towards a particular value.

For simple linearly separable problems this function $a(x)$ might suffice. But even if we have 4 data points this simple model can not optimally solve it for all cases. Therefore, if we can not transform our input data into a better representation, this simple linear model cannot solve a non-linearly separable problem.

Next step might be n^{th} order polynomial function of $a(x)$, and more generally as an infinite series of $a(x)$. We can even go beyond that and represent our model with a non-linear function g of $a(x)$,

$$h(\mathbf{x}) \equiv g(a(x)) = g(\sum_{i=1}^{d} w_i x_i + b). \tag{2.4}$$

Based on this form, we now have a functional $h(x)$ [1] that is powerful but still can not solve some non-linearly separable problems, such as the one in Figure 1. What we will need to do is to combine these functionals to construct even more powerful representations.

1. Functionals are widely used in physics, among which action functional is the most famous one. Action functional is the time integral of Lagrangian, and if we minimize the action, we get equations of motion. This is called the principle of least action, or Hamilton's principle.

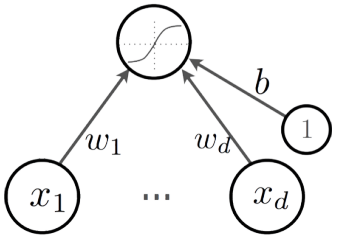

Figure 2.1 – A graphical illustration of an artificial neuron. The input is the vector $\mathbf{x} = \{x_1, x_2, ..., x_d\}$ to which a weight vector $\mathbf{w} = \{w_1, w_2, ..., w_d\}$ and a bias term b is assigned. (Figure adapted from Hugo Larochelle's slides)

2.3.1 Artificial Neuron

The functional $h(x)$ that we constructed can be thought of as an artificial neuron. Figure 2.1 shows us a good illustration of this similarity between biological neurons and our model. We can think of these weight connections as axon-synapse-dendrite connections and construct the whole network of neurons. The function $a(x)$ is called the neuron pre-activation or input activation function. The non-linear $g(x)$ is called the activation function, $\mathbf{w}$ are the connection weights and b is the neuron bias.

2.3.2 Activation Functions

There are various activation functions that we use in machine learning. We will briefly mention only four of those that are widely used in the literature: sigmoid, softmax, hyperbolic tangent (tanh) and rectifier linear unit (ReLU) activation functions.

Sigmoid activation function is a non-linear function where the co-domain is between 0 and 1 with the following form:

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{2.5}$$

It is therefore very appealing to use sigmoid activation function for binary classification problems. If we multiply x with a large constant sigmoid function behaves

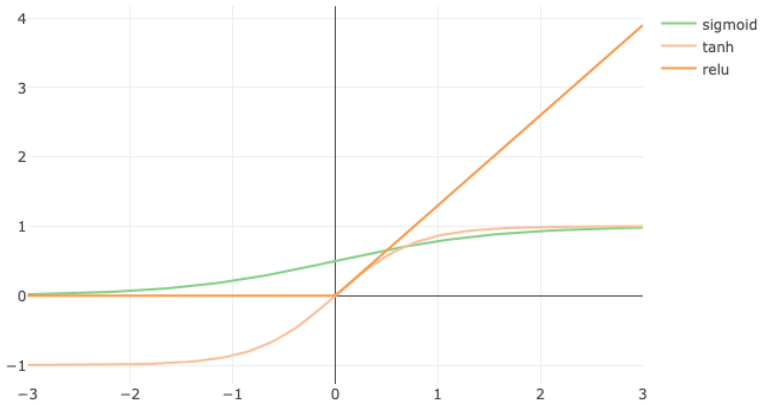

Figure 2.2 – the sigmoid, the hyperbolic tangent (tanh), and rectifier linear unit (ReLU) the activation function

as a differentiable version of Heaviside step function

$$H(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}. \tag{2.6}$$

One can generalize a sigmoid function to multiple dimensions. This generalization is called the **Softmax** activation function, and it has the following form: [2]

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}}. \tag{2.7}$$

The series that we have in the denominator is the normalization factor. It is trivial to see that the softmax activation function is strictly positive and sums up to one. Therefore it is very convenient to use this function for multi-class classification. We can interpret each value as an estimate of the conditional probability of our input being in a particular class, labelled by i among C different classes. And as a result of our model the predicted class will be the one that has highest value of softmax function.

Tanh activation function is a non-linear function where the co-domain is be-

2. Softmax activation has the same form as the Boltzmann distribution, which gives the probability of a particular state determined by its energy. The Boltzmann distribution also has a normalization factor, which is called the canonical partition function. This is done so that we can add up all the terms for possible energy states, which sum up to 1.

tween -1 and 1 with the following form:

$$g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.8}$$

Tanh activation function is useful for sentiment analysis due to its asymptotic behaviour. If we multiply x with a large constant tanh function behaves as a differentiable version of signum function

$$H(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}. \tag{2.9}$$

ReLU activation function is a non-linear function where the co-domain is between 0 and $+\infty$ with the following form:

$$g(x) = \begin{cases} x & x \geq 0 \\ 0 & x = 0 \end{cases}. \tag{2.10}$$

ReLU became the most popular activation function in machine learning algorithms. The main advantage of ReLU over the other two activation functions (sigmoid and tanh) can be seen from Figure 2.2. ReLU has a non-zero derivative for a whole positive range of values, but the derivative of the sigmoid and tanh functions quickly become very small.

2.3.3 Neural Networks

To be able to solve non-linearly separable problems, it will suffice us to take each input of an artificial neuron as the output of various artificial neurons. This representation is called a single hidden layer Neural Network, and it has the schematic form as in Figure 2.3. The functional form of $f(x)$ is given by the following expression:

$$f(x) = o\left(b^{(2)} + \sum_{i=1}^{D_1} \mathbf{w}_i^{(2)} g\left(b_i^{(1)} + \sum_{j=1}^{D_0} W_{i,j}^{(1)} x_j\right)\right). \tag{2.11}$$

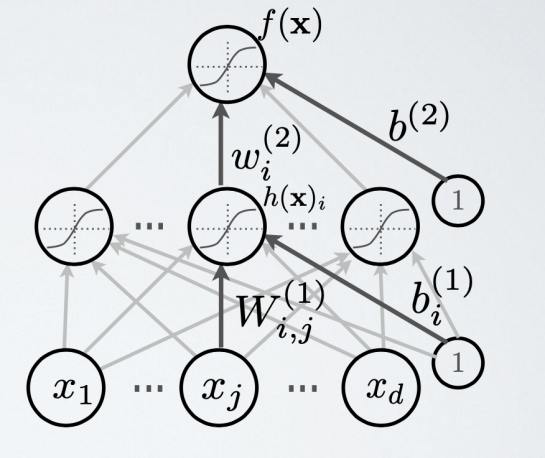

Figure 2.3 – Single hidden layer neural network

The term that appears with the index 1 is the so-called hidden layer pre-activation $\mathbf{a}(\mathbf{x}) = \mathbf{b}^{(1)} + \mathbf{W}^{(1)}\mathbf{x}$. The hidden layer activation is a non-linear function acting on the pre-activation function, $\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$. Notice that $\mathbf{b}^{(1)}, \mathbf{a}^{(1)}, \mathbf{h}^{(1)} \in \mathbb{R}^{D_1}$ are D_1 dimensional vectors. The affine transformation, or in the language of machine learning the weight term is a two-tensor $W_{i,j}^{(1)} \in \mathbb{R}^{D_0 \times D_1} \cong \mathbb{R}^{D_0} \otimes \mathbb{R}^{D_1}$.

One can recursively continue building this model, which will have a larger capacity. This model is called **Multi-layer Neural Network** and has the following mathematical form

$$\mathbf{h}^{(L+1)}(\mathbf{x}) = o(\mathbf{b}^{(L+1)} + \mathbf{W}^{(L+1)}\mathbf{h}^{(L)}(\mathbf{x})), \tag{2.12}$$

for L hidden layers. Let us notice that the final recursion will give us the zeroth level (input) $\mathbf{h}^{(0)}(\mathbf{x}) = \mathbf{x}$. Similar to the previous case $\mathbf{b}^{(L)}, \mathbf{a}^{(L)}, \mathbf{h}^{(L)} \in \mathbb{R}^{D_L}$ are D_L dimensional vectors. And the weight term is a two tensor. Since this affine transformation connects $\mathbf{h}^{(L)}$ and $\mathbf{h}^{(L-1)}$, it is an element of the space which is equal to the tensor product of two vector spaces, $\mathbf{W}^{(L)} \in \mathbb{R}^{D^{L-1} \times D^L} \cong \mathbb{R}^{D_{L-1}} \otimes \mathbb{R}^{D_L}$.

This model is also called Multi Layer Perceptron (MLP) or Fully Connected Network (FCN). Notice that if it were not for the non-linear activations, the full layers would reduce to series of matrices multiplied in sequence. This series of matrices is the same as a single matrix multiplied by the input. This shows us the

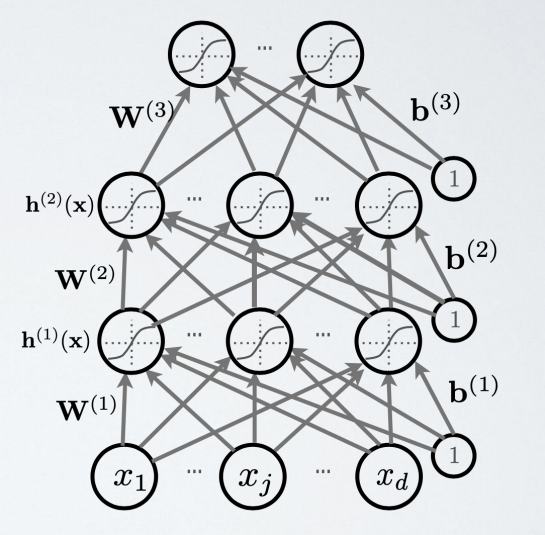

Figure 2.4 – A Three layer neural network. The matrix $\mathbf{W}^{(k)}$ connects the $(k-1)^{th}$ layer to the k^{th} layer and therefore $\mathbf{W}^{(k)} \in \mathbb{R}^{D^k \times D^{k-1}}$ and $\mathbf{b}^{(k)} \in \mathbb{R}^{D^k}$. After each linear transformation (weight multiplication and bias addition), an activation function is applied. (Figure adapted from Hugo Larochelle's slides)

importance of non-linear activation functions to get a large representation.

2.3.4 Universal Approximation Theorem

In this section, we will give some of the results, without proof, related to neural networks' approximation capabilities.

The first theorem for Universal Approximation is limited to sigmoidal functions, by Cybenko (1989). We first define a *sigmoidal function* σ as:

$$\sigma(x) \to \begin{cases} 1 & \text{as } x \to +\infty \\ 0 & \text{as } x \to -\infty \end{cases}$$

Theorem 1. *Let $C([0,1]^n)$ denote the set of all continuous function $[0,1]^n \to \mathbb{R}$, let σ be any sigmoidal activation function then the finite sum of the form $f(x) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w_i}^\top x + b_i)$ is dense in $C([0,1]^n)$*

Therefore the above theorem ensures us that we can always approximate any function arbitrarily close using sigmoidal functions. We can see that the result is for arbitrary width and bounded depth.

Theorem 2. *Any continuous function* $f : [0,1]^n \to \mathbb{R}$ *can be written as*

$$f(x) = f(x_1, .., .., x_n) = \sum_{q=1}^{Z_m} \phi_q \left(\sum_{q=1}^{m} \Psi_q(x_q) \right)$$

The Kolmogorov-Arnold representation theorem (or superposition theorem) **?** was proposed originally to solve Hilbert's 13th problem. But Hecht-Nielsen [19] interpreted Kolmogorov-Arnold representation theorem as a feed-forward neural network, whose activation functions were the inner and outer functions. This would mean that one can represent any continuous function, not with one hidden layer without any approximation, assuming that we can choose the non-linearity of each unit. Girosi and Poggio claimed that this interpretation is irrelevant due to the non-smoothness of inner and outer functions. Kurkova later showed that Kolmogorov's theorem on representations of continuous functions of n-variables by sums and superpositions of continuous functions of one variable is relevant in the context of neural networks. Therefore the exactness is no more, but the approximation is there and can be understood by a constructive proof of The Kolmogorov-Arnold representation theorem.

In machine learning practice, we see deeper networks with a large number of hidden layers. Therefore one would like to know if there exists any universal approximation theorem for arbitrary depth. The following theorem by Lu et al. (2017) states that arbitrary depth universal approximation theorem can be done for L1 distance, ReLU activation function:

Theorem 3. *(Universal Approximation Theorem for Width-Bounded ReLU Networks). For any Lebesgue-integrable function* $f : \mathbb{R}^n \to \mathbb{R}$ *and any* $\epsilon > 0$*, there exists a fully-connected ReLU network A with width* $d_m \leq n + 4$*, such that the function* F_A *represented by this network satisfies*

$$\int_{\mathbb{R}^n} |f(x) - F_A(x)| dx < \epsilon$$

2.3.5 Going Beyond Deep Networks

When the depth of the neural network is larger than 3 it is customary to call it a **Deep Neural Network**. And the intuition behind depth is that at each layer, a

Deep Neural Network (DNN) learns a different level of abstraction to encapsulate the whole level of abstraction.

Based on Universal Approximation Theorem, we see that any input can be represented with an FCN that is deep or wide enough with high accuracy. But the number of weights become so large even for an image that has a small resolution. For an image that has 256x256 pixels, one would need 10^{11} number of components for the weights if we want to keep the first layer of neurons the same as the dimension of the input vector. Therefore it makes sense to use some properties of the data, such as translation symmetry or sequential nature, to come up with better representations.

2.3.6 Convolutional Neural Networks

Convolutional Neural Networks (CNN) is a type of neural network that would avoid the problems that we discussed in the previous section with high-resolution images. Instead of connecting every pixel, which are components of the input vector, to a layer of neurons, it uses the idea of local connectivity. The images in various data-sets that usually have rectangular shapes are scanned with typically a square[3] "kernel." Each scanning (convolution) is a dot product between the values inside the kernel and the pixel values that have the same size. Typically the number of parameters of the kernels is small, but one can choose hundreds of different kernels to increase the capacity of the model at each layer. With the CNN model or so-called "architecture," we can go deeper without immediately reaching billions of parameters.

Another technique that we use in the CNN model is called pooling. This technique lets us reduce the image size by following either a MaxPooling or an Average-Pooling operation. In the former case, We take the pixel with the maximum value in a specific grid and replace the whole grid with a single number. In the latter case, we replace the entire grid with a number equal to the average of the values in the pixel values. Kernels and subsampling methods help us to benefit from local translation invariance of images via parameter sharing.

The CNN model captures the different features of an image with various kernels. We seem to detect similar structures throughout the image, such as edges and lines.

3. There are various types of kernels ranging from a square to a hyper-cube or a deformable kernel cite1904.08755 and 1904.08889

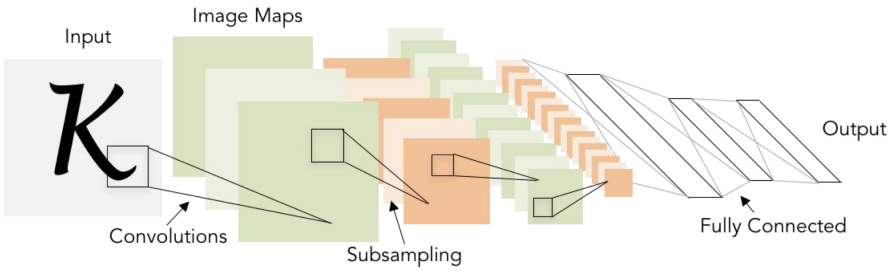

Figure 2.5 – A schematic description of particular CNN model called LeNet-5. (Figure adapted from Fei-Fei Li's slides)

CNNs superiority over MLPs has made it the most popular model used in image recognition, drug discovery and many other areas related to images.

2.3.7 Recurrent Neural Networks

We can make use of the sequential nature of the data when building a machine learning model. There are some quantities, which can be a certain meaning of a sentence, that are common or invariant if we translate from one word to another. This is very similar to the concept of time translation symmetry in physics. Therefore it would make sense to use the same weights while constructing the network. This is the main idea of Recurrent Neural Networks (RNN). Its architecture has the schematic form given by Figure 2.6. It has the following mathematical form,

$$\mathbf{h}_t = \sigma(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b}) \tag{2.13}$$

$$\mathbf{y}_t = (\mathbf{V}\mathbf{h}_t + \mathbf{c}), \tag{2.14}$$

where all the weights $\mathbf{W}$, $\mathbf{V}$ and biases $\mathbf{b}$,$\mathbf{c}$ are the same for all the terms in the sequence.

A problem arises after repeated usage of the same weights. If we look at this from the perspective of linear algebra, we can see the problem. What we do with this RNN model is taking a matrix and multiplying it with itself. By doing that, we force the final result to be dominated by the direction of the largest eigenvalue of the matrix. And the eigenvalues that have small values will cease to contribute, and we will lose useful information.

There are many ways to avoid this problem. One way of the earliest model which

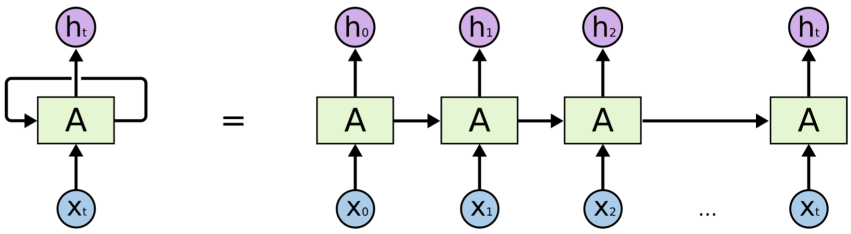

Figure 2.6 – A schematic description of Recurrent Neural Networks (Figure adapted from Chris Olah's Tutorial)

avoids this problem is called LSTM networks. There are input forget and output gates inside each memory cell, which lets some of the information get through. More recently, more advanced techniques were developed to avoid this problem, such as Attention Mechanism (Bahdanau et al., 2015) and Transformer Networks (Vaswani et al., 2017). With these recent advances, we can train extremely large models with massive success in language modelling (Brown et al., 2020).

2.3.8 Generative Models

Generative models can be divided into two categories, prescribed and implicit generative models (Diggle and Gratton, 1984). For the case of prescribed generative models, we explicitly specify the data distribution that has tractable likelihood functions for a random variable, with some parameters θ. Then we learn θ using a log-likelihood estimation on the observed data. On the other hand, we do not provide an explicit specification of any distribution for the case of implicit generative models. We define a stochastic process that generates (simulated) data from an underlying data distribution.

Sigmoid belief networks (Neal, 1992), latent Dirichlet allocation models (Blei et al., 2003) and Variational Auto-Encoders (Kingma and Welling, 2014) are examples of prescribed generative models. Generative Adversarial Networks (GANs) Goodfellow et al. (2014) are an example of implicit generative models.

2.4 What to Learn

We have shown that different types of deep neural network models can represent the input data differently. We have also seen that the Universal approximation theorem ensures that DNNs can approximately represent any function. This raises an important question: How do we measure the success of our model? In this section, we will first define what success or conversely failure is in the context of learning models.

2.4.1 Learning vs Memorization

The essential point in machine learning is learning with some data at hand and then using that knowledge to do an excellent job on the unseen data. Therefore its objective is not merely optimizing a function to represent an input data but rather a generalization.

The strategy to get a good generalization performance for a machine learning model is following a four-step procedure:

1. Divide the full data into three parts, training, validation and test set.
2. Build a model/network with some parameters and use a particular optimization procedure to "learn" these (trainable) parameters using the training set.
3. Gauge the success of this model with the validation set. Either change certain parts (called hyper-parameters) of the network architecture or some other parameters of the model that do not change during step-2.
4. Leave some part of the data as a surrogate of "unseen" data to measure the final success of your model.

Let us now discuss the issue of measuring the success of a learning model. We will assume that the dataset is independent and identically distributed (IID) and consists of n number of d-dimensional input vectors:

$$S = (Z_1, Z_2, ..., Z_n)\,, \tag{2.15}$$

The dataset is collected from some unknown distribution $P(Z)$. Z_i here is either $\mathbf{x}_i), y_i$ pairs for labeled data or just $\mathbf{x}_i$) for unlabelled data. The learning tasks that we discussed at the beginning of the chapter are related to following problems:

— Classification task $f_\theta : \mathbb{R}^D \to \{1, ..., m\}$.
— Regression task $f_\theta : \mathbb{R}^D \to \mathbb{R}$.
— Probability density estimation task $f_\theta : \mathbb{R}^D \to \mathbb{R}^{>0}$.

Here m is the number of classes, and θ represents the set of parameters of the model f that is used for the task. For these tasks, we would like to find the difference between our prediction and the ground truth. This difference $L(f_\theta(\mathbf{x}), y)$, which gives us the measure of the success or rather the failure of our model, is called the **Loss Function**.

The functional form of loss function for the regression task is either the quadratic error

$$L(f_\theta(\mathbf{x}), y) = (f_\theta(\mathbf{x}) - y)^2, \tag{2.16}$$

or the absolute loss

$$L(f_\theta(\mathbf{x}), y) = \|f_\theta(\mathbf{x}) - y\|. \tag{2.17}$$

For binary classification tasks one usually uses the binary cross-entropy loss function

$$L(f_\theta(\mathbf{x}), y) = -y\, log(f_\theta(\mathbf{x})) + (1 - y)\, (log(1 - f_\theta(\mathbf{x}))) \,. \tag{2.18}$$

For multi-class classification tasks, we use a softmax activation function for the last layer. And therefore, the result of our model will give m number of outputs; each corresponds to the probability of the input is in a particular class. The corresponding cross-entropy loss function is equal to the negative of the total log-likelihood contributions coming from each class

$$L(f_\theta(\mathbf{x}), y) = \sum_{i=1}^{m} -log(f_\theta(\mathbf{x}_i)). \tag{2.19}$$

The final type of loss function is related to unsupervised learning tasks. To measure how successful our models are in producing similar distributions, we usually end up having to compare two distributions. This is done by measuring the statistical distance between two distributions with various metrics. The most popular metrics that we use are:

— Kullback–Leibler (KL) divergence
— Jensen–Shannon (JS) divergence
— Wasserstein distance

Definition 4 (Kullback–Leibler divergence)**.**

$$KL(P\|Q) = \int p(x)\, log\left(\frac{p(x)}{q(x)}\right) \tag{2.20}$$

Jensen–Shannon divergence is simply the symmetrized version of KL divergence,

Definition 5 (Jensen–Shannon divergence)**.**

$$JS(P\|Q) = \frac{1}{2}KL(P\|R) + \frac{1}{2}KL(Q\|R), \tag{2.21}$$

where $R \equiv \frac{1}{2}(P + Q)$.

One can also give a precise definition of Wasserstein distance, but in order to avoid more mathematical formulation, we will briefly discuss the intuition behind it. In the context of transport theory, we take all possible ways to transport a distribution P to Q. Then we calculate the integral of the total distance travelled along the x axis with respect to x. The minimal total distance that we get among all possible ways is called the Wasserstein distance.

2.4.2 Empirical Risk Minimization

We now know the learning algorithm's objective, find the best function with the best parameters that give a minimum total/average loss for the whole data set. This is called generalization error or expected risk.

Definition 6 (Generalization Error)**.**

$$R(f) \equiv \mathbb{E}_{p(\boldsymbol{x},y)}[L(f_\theta(\boldsymbol{x}), y)] \tag{2.22}$$

In practice we can not calculate the expectation since $p(\mathbf{x}, y)$ is unknown. Instead of the expected risk, we calculate the average error over some finite data set S that is a subset of $p(\mathbf{x}, y)$. We call this approximate risk, the Empirical Error

Definition 7 (Empirical Error)**.**

$$\hat{R}_S[f] \equiv \frac{1}{n}\sum_{i=1}^{n} L(f_\theta(\boldsymbol{x}_i), y_i) \tag{2.23}$$

The difference between these two errors is called the **Generalization Gap**.

Therefore the best thing that we can do, with a limited sized data set, is minimizing the empirical error instead of the true risk. We call this objective the Empirical Risk Minimization (ERM).

However, there is an essential caveat to the ERM picture. Minimizing just the empirical error (training error) is not the right way to estimate our model's success. Because our model will be too specific for the data we have, and it is doomed to underperform on unseen data. Therefore it is crucial to leave some part of our data as a test set to gauge the success of our model's performance. ERM is a good estimator for our model's success if and only if we get small test error and small training error. We will discuss these issues more, basic techniques and best practices during the learning procedure in the next sections.

2.5 How to Learn

This section will answer the next important question during the learning stage: How do algorithms learn?

In the context of empirical risk minimization our objective is finding the parameters θ^* of our model that minimizes the average training loss

$$\theta^* = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(\mathbf{x}_i), y_i). \tag{2.24}$$

Since $\boldsymbol{\theta}$ is a d dimensional vector minimizing the loss function with respect to $\boldsymbol{\theta}$ is done by taking the gradient and equating each component to zero. In principle, one has to solve the analytical solution can be found by solving d equations

$$\nabla_\theta \hat{R}_S[f_\theta] = 0 \rightarrow \begin{cases} \frac{\partial \hat{R}_S[f_\theta]}{\partial \theta_1} & = 0 \\ \frac{\partial \hat{R}_S[f_\theta]}{\partial \theta_2} & = 0 \\ & \vdots \\ \frac{\partial \hat{R}_S[f_\theta]}{\partial \theta_d} & = 0 \end{cases}$$

But only for very simple models it is possible to solve these equations analytically. For high-dimensional complicated models, these equations can only be solved using iterative algorithms.

The positive side of numerical methods is that we can use powerful computational tools at our disposal. The negative side of numerical methods is being blindsided. If we think about a landscape with millions of parameters/dimensions, we will have a vast number of local minima and saddle points. The computational power that is needed to solve this highly non-convex optimization problem was not possible until recently.

2.5.1 First Order Optimization

The first type of iterative method that we will discuss is the **Gradient Descent**. Before giving an algebraic derivation, let us discuss the geometrical intuition behind the gradient. If we think of the loss landscape, one should follow the direction where the value of it decreases, which is the same as the opposite to the gradient direction. More formally, let us assume that we would like to go iteratively towards the minimum of a function $f(\mathbf{x})$ by taking a step of size $\Delta\mathbf{x}$. If Taylor expand $f(\mathbf{x})$ at its new position $\mathbf{x} + \Delta\mathbf{x}$, we get the following condition for descent

$$f(\mathbf{x}_{k+1}) = f(\mathbf{x}_k + \Delta\mathbf{x}) \approx f(\mathbf{x}_k) + (\Delta\mathbf{x})^T \nabla f(\mathbf{x}_k) < f(\mathbf{x}_k), \tag{2.25}$$

since after the kth step, the value of the function should be less than what it was in the previous step. In order to ensure that the last inequality holds we can choose

$$(\Delta\mathbf{x})^T = -\eta \nabla f(\mathbf{x}_k)^T, \tag{2.26}$$

where η is some positive constant.

To sum up, the gradient descent algorithm tells us to iteratively replace the position's value with itself minus the gradient of the function at that position

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta \, \nabla f(\mathbf{x}_k). \tag{2.27}$$

The constant η is called the learning rate since it controls the size of the step from its current position. It is customary to decrease the learning rate with each

iteration, and to find the best practices for learning rate schedules is an active research topic Smith et al. (2018). In the context of ERM picture, the update formula for the model parameters is

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \nabla \hat{R}_S[f_{\boldsymbol{\theta}_k}]. \tag{2.28}$$

Another important issue about the learning procedure is deciding when to stop training. We would stop training when the model parameters cease to change with consecutive epochs. This condition corresponds to

$$\| \frac{\partial \hat{R}_S[f_\theta]}{\partial \theta} \| < \delta \tag{2.29}$$

where delta is called the patience parameter. But one usually stops earlier to avoid over-fitting. We will discuss these issues in detail in the following sections.

In the gradient descent algorithm, we update the parameters by subtracting the gradient of the loss function. We defined the objective as taking the gradient of the empirical error $\hat{R}_S[f_\theta]$. Taking the gradient of the mean of losses is equivalent to taking the mean of the gradients of each loss

$$\frac{\partial}{\partial \theta} \hat{R}_S[f] = \frac{\partial}{\partial \theta} \frac{1}{n} \sum_{i=1}^{n} L(f_\theta(\mathbf{x}_i), y_i) = \frac{1}{n} \sum_{i=1}^{n} \frac{\partial}{\partial \theta} L(f_\theta(\mathbf{x}_i), y_i) \tag{2.30}$$

In the above equation, n is the number of inputs that we take before updating the parameters of our model. We can take this to be the whole training data set. This choice is called the **Batch Gradient Descent**. It will give us a good sense of the direction towards an optimal position, but it comes with a cost, long training time.

The other extreme is taking $n = 1$ limit, i.e. updating the model parameters after each single input. This optimization method is called **Stochastic Gradient Descent** (SGD), and it is one of the most popular choices for optimization in machine learning. Since it will collect noise due to collecting gradient for small sample sizes, SGD still generalizes well (Keskar et al., 2017).

There is a midway between these two extremes, that is **Mini-Batch Gradient Descent**. With this method, we use small parts/mini-batches of the whole data set. We use the advantage of the GPU's parallel computation ability and more

frequent updates of model parameters compared to the whole batch training.

There are various first-order optimization algorithms that depend on the gradient of the loss function and do not depend on higher-order derivatives. Most of them use a simple idea of momentum inspired by physics. One uses the momentum towards the minimum of the loss function when updating the model parameters. Some of these momentum-based algorithms are Nesterov, ADAGRAD, RMSProp and ADAM. They differ in various aspects, such as calculating the gradient of the loss function before or after the velocity/momentum parameter update. Adapting the learning rates and using ideas, such as exponential moving averages, are all shown to help faster convergence.

To sum up, vanilla SGD, SGD with momentum and ADAM are the three most popular optimization algorithms that are used in machine learning. Choosing among the whole class of optimization algorithms is a difficult task, and the performance of each method varies with the specific problem at hand. Before making this important choice for a task, it is always helpful to use standardized benchmark results (Schmidt et al., 2020).

2.5.2 Second Order Optimization

We can go beyond the first order in Taylor series in eq.2.25, and go up to second order in $\Delta\mathbf{x}$. If we then minimize this approximation with respect to the perturbation parameter $\Delta\mathbf{x}$ we get the following condition

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \mathbf{H}^{-1}\nabla_{\boldsymbol{\theta}}\hat{R}_S[f_{\boldsymbol{\theta}}]. \tag{2.31}$$

In the above equation $\mathbf{H}$ is called the Hessian matrix and is a square matrix where the elements are partial derivatives of with respect to the relevant variables. Here its elements are expressed in terms of model parameters as

$$H_{ij} \equiv \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}. \tag{2.32}$$

The optimization method given in eq. 2.31 is called Newton's Method. Compared to first-order optimization methods, this is computationally expensive. However, there are quasi-Newtonian methods, which are approximations to the inverse of the Hessian matrix. But these approximations work better when the Hessian

changes adiabatically. Broyden-Fletcher- Goldfarb-Shanno (BFGS) method is the most widely used technique among quasi-Newton algorithms.

Due to the quadratic memory requirement of BFGS algorithm, Nocedal (1980) proposed Limited-memory BFGS. Both Newtonian and quasi-Newtonian optimization methods are more powerful compared to gradient descent algorithms. But computation costs of these algorithms make them non-practical for machine learning models.

Finally, one can meta-learn the parameters of the optimizer without the user explicitly specifying them Metz et al. (2020). It is also claimed that these meta-learned optimizers show evidence of better generalization for the data which do not come from IID, called out-of-distribution.

2.5.3 Initialization

Weight initialization can also be an important issue during the learning stage. This can happen when one uses the tanh activation function, which is very large in certain regions. And for RNN style architectures, this can result in two possible problems: vanishing or exploding gradients.

For exploding gradients, the gradient clipping technique can be used. Gradient clipping is a rescaling procedure of the gradients of the loss function when the norm of it exceeds a certain threshold. The rescaling is done before updating the parameters, which avoids exploding gradient problem.

The vanishing gradient problem is harder to solve compared to exploding gradients in the case of RNNs. There are two different approaches to solve this problem. One is using architectures such as LSTMs, ResNET (He et al., 2016) or DenseNET (Huang et al., 2017) style networks. LSTM uses input and forget gates to control the flow of information. For ResNET or DenseNET style networks, one uses skip-connection methods to ease gradient flow between various layers, which are not just consecutive.

The other approach to solve the vanishing gradient problem is weight initialization. Glorot and Bengio (2010) provided an initialization scheme, with their seminal work, for tanh activation functions. Later He et al. (2015) was able to provide a successful initialization scheme for ReLU activation functions.

Recently Ioffe and Szegedy (2015) were able to develop a method that resulted

in more stable and faster training. They were able to do it by merely normalizing each layer's inputs by relocating at/around the center and usually close to unit variance. This method is called Batch Normalization. More recently, Yang et al. (2019) showed that one could get good results and even with very deep networks using skip connections.

2.6 Regularization

The main objective of machine learning is minimizing generalization error, as we briefly mentioned in the previous sections. In this section, we will discuss the issue of regularization, which is a method to improve the generalization performance of a learning model.

2.6.1 Bias variance trade-off

In machine learning, choosing a model is the first step to take after collecting and analyzing the data. One of the essential questions that face us is how complex/rich our model is going to be. And also, how do we define the richness of a model.

The richness/capacity of a model is related to the number of free parameters; there are various ways to measure the complexity of a learning model. Rademacher complexity and Vapnik–Chervonenkis dimension of Vapnik (1971) are just two different formal ways to measure the capacity of a model.

The answer to the question of deciding on the level of complexity of a model is very important if we want our model to achieve a low generalization error. In order to do that, we need to avoid two extremes: under-fitting and over-fitting. Under-fitting merely is choosing a model/hypothesis among a set of all possible functions/hypothesis class that is too simple for a given task. A linear model for image classification can be given as an example. When we use "simple" functions, variance becomes small, and our hypothesis turns out to be biased.

The other extreme is choosing a model that is too rich for a specific task. In the case where the "capacity" of our model is high, we will do a very good job on the training set. But that does not mean that our model will do a good job on the

test set. When we get very high accuracy on the training set a low accuracy for the test set, we call this over-fitting.

This conflicting behaviour of bias and variance is called bias-variance trade-off. The most popular strategy to follow to get good generalization performance is by increasing the complexity of our model until it reaches a point where the validation accuracy starts decreasing with an increasing training accuracy.

This classic bias-variance trade-off behaviour also an active area of research. Recently Belkin et al. (2019) suggested empirical evidence for "double-descent" behaviour instead of the classic U-shaped bias-variance trade-off curve of generalization error. They claimed to provide evidence for the existence of a double descent curve for various models and datasets.

2.6.2 Regularization

The motivation behind regularization is to reduce the variance of the selected model but keep bias under control by putting extra constraints on the model. We can call our method as the Structural Risk Minimization. The idea is minimizing the so-called structured risk

$$\tilde{R}_S[f_\theta] \equiv \hat{R}_S[f_\theta] + \beta\Omega(\theta), \tag{2.33}$$

where the first term is the expected risk of the training set, $\Omega(\theta)$ is the regularization function. The constant β term is a hyper-parameter that we choose to control the effect of the regularizer.

If the form of the regularization function $\Omega(\theta)$ is the square of the $\mathbf{L^2}$**-norm**, we call it **Ridge regression**. If we choose $\Omega(\theta)$ to be $\mathbf{L^1}$**-norm** we call that regularization scheme, least absolute shrinkage and selection operator **Lasso**

$$\Omega(\theta)_{\text{Ridge}} = \frac{1}{2}\|\theta\|_2^2 \qquad \Omega(\theta)_{\text{Lasso}} = \|\theta\|_1 \,. \tag{2.34}$$

One can show that ridge regression is equivalent to imposing a Gaussian prior over the model. Ridge regression effectively results in the decaying of the weights of the model. Therefore $L2$ regularization is sometimes called weight decay.

Early stopping is also used as a regularization technique. One can look at the parameter space of the model and show that early stopping has the same effect as

ridge regression. This comes due to the fact that both of these methods restrict the volume of this space once the initial parameters are set.

Another useful regularization technique that is used is called Dropout. It is implemented by randomly deleting some hidden nodes with a given probability p, which itself is a hyper-parameter. Srivastava et al. (2014), in their work, claimed that dropout reduces over-fitting by preventing co-adaptation.

Recently Zhang et al. (2017) suggested that we do not need explicit regularizers like dropout and weight-decay for good generalization performance. They claimed that SGD acts as an implicit regularizer. Neyshabur et al. (2015) claimed that one could characterize implicit regularization with minimization of norms, but more recently, this claim was challenged by Razin and Cohen (2020).

3 Experimental Results

In this chapter, we will discuss the choice of architecture in order to solve the electron identification problem. Then we will present our results and compare them with the performance of the traditional identification methods. Finally, we will go over possible directions to proceed to improve our results.

3.1 Choice of Architecture

Choosing an architecture is an important part of any successful machine learning algorithm. We need to analyze our data and then decide which types of networks could be an optimal choice. For the EL-ID task, our data have four distinctive parts:

— Six sequential 7x11 pixel sized images
— One 56x11 image
— Tracks
— Scalars

In principle, we can feed all different parts of our data into simple FCNs and then combine each output into a final FCN. This would work, but we would not be using any advantage of the physical structure of our data.

First of all, we know that the success of CNNs for images is well-established. Therefore, it would make sense to use CNN architecture for the calorimeter images. We can, in fact, use CNNs for each image and FCN for the tracks and the scalars and a final FCN at the end. But in doing so, we would not be using the sequential nature of the six images that we have in our dataset. We will discuss how to exploit this property in the next section best.

Tracks are spatio-temporal quantities; therefore, in order to make use of this property, it might be better to choose a different architecture over FCN. It turns

out that the best performance was achieved when we convert the 15 track points for each electron and five variables related to them to a 15x5 pixel image and feed them to a CNN with a (1x1) kernel. This success could be due to CNNs ability to capture spatio-temporal quantities, even though the tracks are not images.

The final part of the data, i.e. scalars, is not sequential or does not have other common features that we can exploit. Therefore we did not have any a priori reason to choose another model over an FCN. But we observed an increase of accuracy when we used the quantiles transform technique for the scalars. This is a useful pre-processing method and reduces the effect of outliers. It does it by transforming the inputs to have a normal probability distribution.

3.2 Sequential Images

As we mentioned in the previous section, the main uncertainty comes with making an architectural choice to represent six consecutive images. Having sequences of images, it would make sense to look at this problem as a video classification task.

Similar to a natural language processing task, video classification problems can be solved by simply feeding outputs of CNNs as an input to RNNs at each step. Donahue et al. (2015) used this simple idea to solve a visual recognition and description task using RNNs. Similarly Srivastava et al. (2015) used LSTMs to learn representations of video sequences. Shi et al. (2015) extended the fully connected LSTM to have convolutional structures (ConvLSTM) in both the input-to-state and state-to-state transitions to solve weather forecasting problem. ConvLSTM is one of the state of the art architectures that are used to capture spatio-temporal structures.

Although we made an analogy between the video classification task and these images, the similarity goes only so far. A video is a collection of thousands of images, but here we have only six. Therefore using LSTM type of models that are designed to capture features of long sequences would not be an appropriate choice. Choosing a relatively more "static" architecture might be better. Karpathy et al. (2014) used two-dimensional CNNs to solve a video classification problem with 1 million YouTube videos that belonged to 487 classes. Similarly, Yao et al. (2015) used 3D CNNs for video description problem. Therefore, it was shown that one

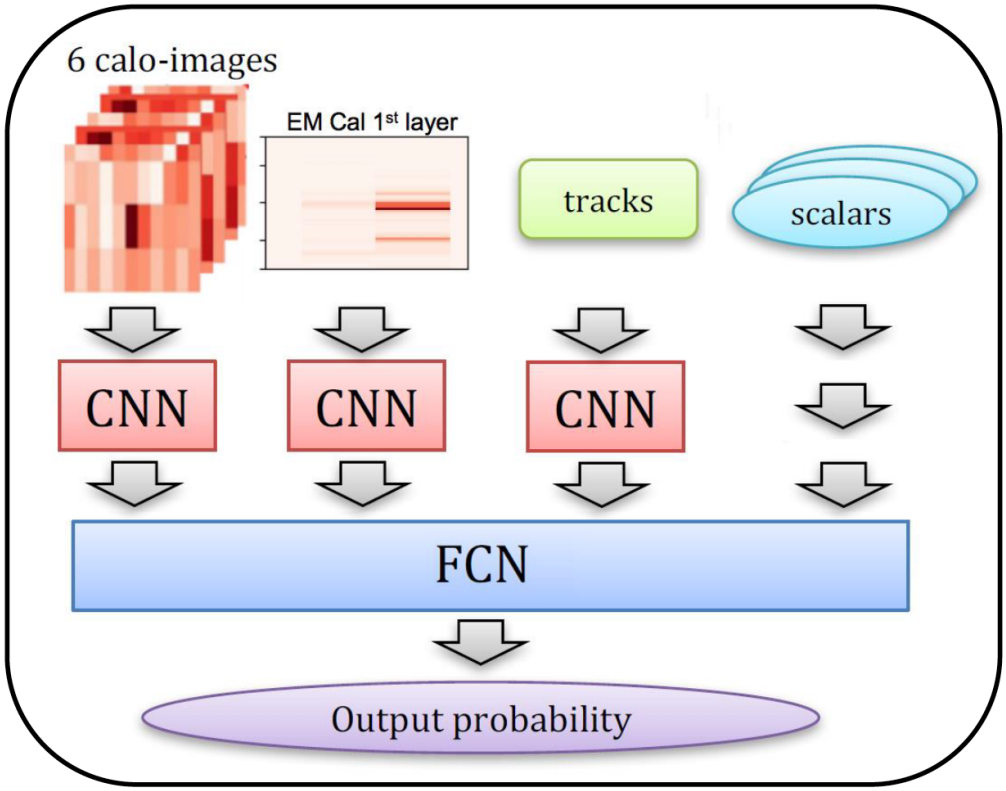

Figure 3.1 – The Architecture for EL-ID task (Image by Kazuya Mochizuki)

can capture spatio-temporal features by "static" architectures. It turned out that choosing static 2D or 3D CNNs worked better for our EL-ID task since it is a very short sequence.

For the usual coloured images, one would feed images that come from three different channels (Red, Green and Blue). Here we have six images that have only one colour, and we feed these six images from 6 different channels to a CNN. The architecture that we chose to use (Mochizuki, 2020) that has the properties that we mentioned can be seen from Figure 3.1.

3.3 Experimental Results

After making an architecture choice, the next step is to start training the model and find out the best hyper-parameters. At the early stages of the EL-ID task, we

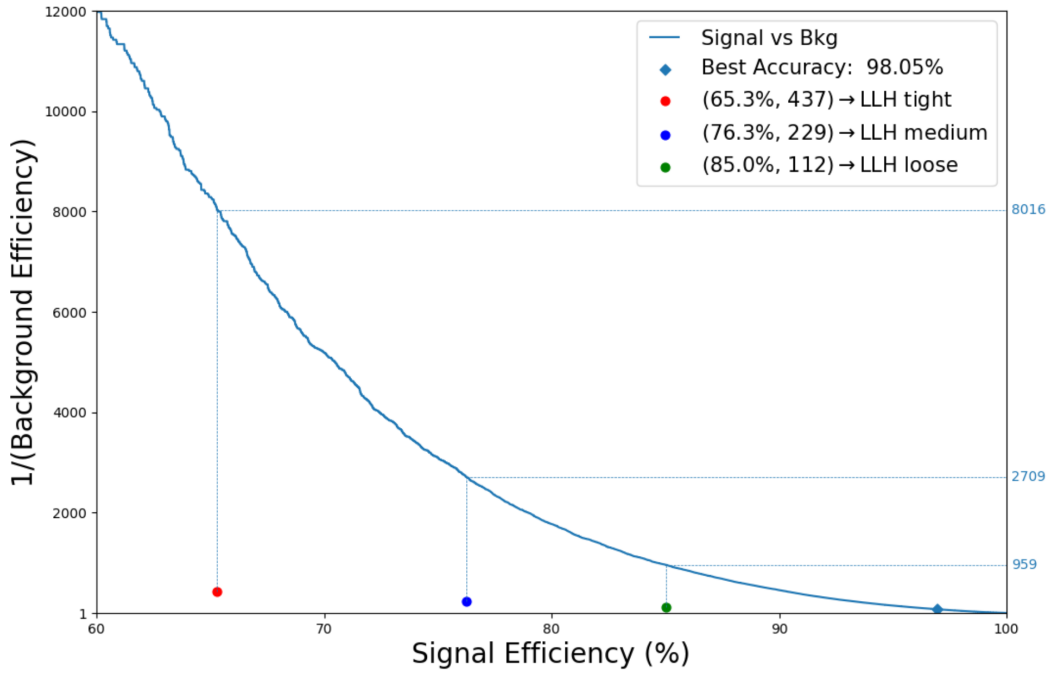

Figure 3.2 – ROC Curve for the best result with 14M electrons

had a relatively small number of images. Later on, we were able to find more optimal ways to produce data. Therefore we were able to increase the number of images by almost three orders of magnitude, from 600 thousand to 100 million images.

In Figure 3.2, the performance of our model is shown with Receiver Operating Characteristic (ROC) curve. ROC curve is a plot that shows the quality of models' capacity to solve a binary classification task when its threshold changes. When we try to distinguish two distributions (isolated electrons vs background electrons), four different cases occur True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The standard ROC curve is plotted using two quantities

$$\text{Background Rejection} \equiv \frac{TN}{TN+FP}\ ,\ \text{Signal Efficiency} \equiv \frac{TP}{TP+FN}. \tag{3.1}$$

When TN >> FP background efficiency approaches to 1. In that case, any improvement of our model will make it even closer to 1, but it will be very difficult to notice from the ROC curve. Due to this reason, we plot the ROC curve as

1/(background efficiency) vs signal efficiency

$$1/(\text{Background Efficiency}) \equiv \frac{FP+TN}{FP}. \tag{3.2}$$

For the EL-ID task, ATLAS uses the LH method, and it gives three operating points to quantify the background rejection power of the method: Loose, Medium, or Tight. These points are in order of decreasing electron efficiency and increasing background rejection. And we use simulators to determine these values of three operating points.

From Figure 3.2, we can see that our model performs 8, 12 and 18 times better than the operating points of LH estimates. In short, the performance of our deep learning model surpasses the current model that ATLAS uses by a huge margin for the EL-ID task.

Table 3.1 – Best hyperparameters for EL-ID Task

Method/Part of Architecture	Search method	Best Values
Optimization	ADAM, lr	1e-3
Regularization	L2	1e-7
Regularization	Dropout	0.1
CNN Images	Kernel Size &Padding & Stride	(3x3, 1, 1)
CNN Tracks	Kernel Size &Padding & Stride	(1x1, 0, 1)
CNN	Number of features	32
CNN	Number of Conv2d layers	3
CNN	Number of FCN layers	3
CNN & FCN	Number of Neurons	100

One important question is if each part of the data was instrumental in achieving high accuracy, or could we get rid of some part of the data and still get comparable results. In order to answer that question, we did experiments where we used each part of the data separately. Our results showed that none of the parts of the data was itself enough to give 98.05% accuracy. As one can see from Table 3.2, the scalars were the most useful part of the data, and the tracks are the least useful one. But we were able to achieve the best results only when we included all parts of the data, including the tracks.

Among all popular optimization methods that we tried, ADAM was the best optimizer. Although we were able to get comparable results with SGD+Momentum,

Table 3.2 – Accuracy results using only one part of the data

Scalars	7x11 Images	56x11 Images	Tracks
97.2%	96.4%	95.2%	91.1%

the training time for it was longer than ADAM. The best learning rate was 1×10^{-3} but decreasing it towards the end of the training up to 1×10^{-6} gave not appreciable but slightly better results. But using various learning rate scheduler methods that were available in libraries did not improve the performance.

Finally, we investigated the effect of regularization. A very small L2 regularization 1×10^{-7} helped to get slightly better results. And our hyper-parameter search showed a dropout of 0.1 was the best value that we used after ReLU activations both for FCN and CNN parts of our model.

3.3.1 Architectural Trials

At the early stages of our work, we used CNN+RNN architecture due to the sequential nature of calorimeter images. We could not outperform an LH estimate and could not get an accuracy of more than 80%. Then we decided to used the stacked 2D CNN architecture and were able to get close to 90% accuracy levels. Therefore before being able to increase the data size, we decided to optimize our model based on stacked consecutive images that were fed to the 2D CNN model from 7 "colour" channels.

Later we connected this CNN model with other Neural Networks that we used to represent the tracks and scalars. This is similar to two-stream models that are used for video recognition (Simonyan and Zisserman, 2014). We used standard FCNs for scalars but used the quantiles transform technique for the inputs. And finally, we obtained better results when tracks were converted to 5x13 images that are fed to the CNN model compared to just inputs fed to a standard FCN. As we discuss in the previous section, we attribute this to the spatio-temporal nature of the tracks and CNNs ability to model these features (Karpathy et al., 2014).

After achieving much better results compared to the LH estimate, we checked if there was any room for further improvement. This is typically done in machine learning tasks by making the model deeper or wider. In order to do that

we tried three different architectures, ResNet style (He et al., 2016), WideResNet (Zagoruyko and Komodakis, 2016) and DenseNet (Huang et al., 2017) style architectures.

To complete these trials at an optimal time, we used these models only for the images to check if there is an increase in accuracy. None of these models showed an increase even though the number of parameters two orders of magnitude higher than simple CNNs. The training time increased with more parameters than we expected.

Our interpretation of this result is that there is not much to learn by going deeper in this model for our task. This makes sense since the structures of the images that we see in Figure 1.6 are not extremely complicated. Therefore shallow networks would be able to capture the spatial shapes of these figures.

We also implemented CoordCONV architecture (Liu et al., 2018) to improve the performance of our model. CoordCONV is a simple model where one concatenates two more images (one for x and one for y) to the existing image. The pixel values of the extra images are the Cartesian coordinate values of x and y coordinates. Liu et al. (2018) showed that certain RL tasks benefit from CoordCONV layers, and a Faster R-CNN detection model trained on MNIST detection showed 25% better intersection over Union for object detection. For our task, CoordCONV layers did not help to get better results and even led to a small decrease (1%) in accuracy.

3.4 Future Research Directions

In this section, we will go over possible future research directions, some of which are already being pursued by the EL-ID task group:

— Experimenting with various architectures
— Producing more data
— Hyper-parameter scan
— Multi-class classification
— OOD generalization

3.4.1 Experimenting with various architectures

As we mentioned at the beginning of this chapter Yao et al. (2015) used 3D CNNs for video description problem and showed that this architecture better captures spatio-temporal correlations compared to 2D CNNs. This line of work is currently being pursued by the UdeM EL-ID group. The difficulty of this line of work is the number of possible 3D kernels that one can use. Therefore, the grid search is very time-consuming.

One can go beyond using stacked 3D kernels and implement Two-Stream ConvNet models that are used for action classification tasks (Simonyan and Zisserman, 2014; Feichtenhofer et al., 2016; Carreira and Zisserman, 2017).

Another future direction is using the Transformer model, which was introduced by Vaswani et al. (2017). Transformers are widely used to solve NLP tasks, but it is also used for action recognition task by making use of features from the spatio-temporal context (Girdhar et al., 2019).

Even more recently, the so-called Vision Transformer (ViT) model was developed for image recognition tasks (Girdhar et al., 2019). The novelty of this work is that CNNs were not used at all. Instead, pure transformers were applied to patches of images. After being pre-trained on large amounts of data ViT model got better than the state of the art results on many datasets. Therefore we can completely get rid of CNN architecture and implement a ViT type of model for the EL-ID task.

3.4.2 Producing more data

Another direction that would certainly be useful is producing more data. Producing and handling an amount of data that is more than what we already have is a challenging task. But doing that would certainly help us to get better results. Very recently, Dominique Godin, who is a member of the EL-ID task group, was able to produce MC data for 120 million events. This amounts to more than a billion images, and preliminary results were already promising as we expected.

One interesting thing to check is if the performance of our model obeys a certain scaling law as a function of data (Henighan et al., 2020). If our model's performance obeys certain empirical scaling laws, that will give us a chance to adjust our compute budget before producing much more data.

3.4.3 Hyper-parameter scan

As it is the case for all machine learning experiments, doing a thorough hyper-parameter scan by grid-search is a very important part of the whole pipeline. It is currently being pursued by Sergey Panitkin who is a member of the EL-ID task group. He is making a hyper-parameter scan using the Summit Supercomputer of The Oak Ridge Leadership Computing Facility.

3.4.4 Multi-class classification

As we mentioned in the previous chapter, knowing the type of background is a useful information to optimize the data analysis downstream. Therefore making a successful multi-class classification will further improve our ultimate goal of isolating the signal electrons. This is a relatively difficult task, since our dataset is a highly imbalanced. Dominique Godin and Kazuya Mochizuki, who are both members of the EL-ID task group, are working on this task.

3.4.5 OOD generalization

During our experiments, all of the data that we used were produced by MC based simulations. The simulation is able to model SM of particle physics as well as the detector at a very good level. But no matter how good it is, there are still many approximations that are routinely being made during this data producing stage. These approximations are sometimes due to our incomplete understanding of particle physics or sometimes our inability to calculate some higher-order terms that appear in perturbative expansions.

Because of these reasons, one can predict that the real data distribution and the one that we have from the simulations will not be identical. Therefore, it would make sense to make the model so that it would also do a good job on out of distribution (OOD) tasks.

Understanding the underlying principles of OOD generalization performance and how to improve it is a subject that is an active research topic. In the next chapter, we will briefly discuss some methods to improve OOD generalization performance through data augmentation. But our main focus will be the answer to the following question: *If we choose an algorithm that puts us in a* ***wide flat***

minimum *region of a loss landscape, will this help for OOD generalization performance?* We will present some algorithms that claim to take us towards these regions. Finally, we will show the results of our implementation of these algorithms to the EL-ID task and try to answer the above question.

4 OOD Generalization and Flatness

In this chapter, we will discuss the concept of sharpness and wide flat minimums of the loss surface in detail. Then we will make a conjecture about the relationship between wide flat minimums and OOD generalization performance. Finally, we will implement two algorithms to different parts of our EL-ID dataset based on their angular locations and use that to mimic domain-shift behaviour. We will then discuss our results in the context of our OOD Generalization vs Flatness conjecture. Finally, we will end with future work to be done.

4.1 Sharpness based complexity measures

The concept of sharpness of local minimum has been introduced in the Machine Learning literature in the early and mid 90th in the works of Hinton and von Cramp (Hinton and von Cramp) and Hochreiter and Schmidhuber (1995). Both works motivate the advantage of flat optimum over the sharp ones through the minimum description length principle: sharp minimum requires higher precision in their specification, whereas flat minimum can be given with less precision or fewer bits of information, thus resulting in a less complex network and better generalization performance. Intuitively, low necessary description length suggests robustness to noise in the parameter space and thus lower expected overfitting. One could perturb the weights at the optima a bit without effectively increasing the loss value.

In the following years, this concept has received a lot of attention in the research literature. Most of the work is based around the simple *empirical* observation - local minima that generalize well tend to lie in 'wide valleys' of the loss landscape, rather than sharp, narrow ones. Related work can be roughly divided into the following two categories: works discovering new properties of loss-landscape geometry both

theoretically and empirically (Petzka et al., 2020; Achille and Soatto, 2018a; Dinh et al., 2017; Liang et al., 2017; Keskar et al., 2017; Sagun et al., 2017; Jastrzebski et al., 2018; Jiang et al., 2019) as well as works introducing concrete algorithms for reaching solutions with particular properties, such as wide and flat regions or robust feature representation (Chaudhari et al., 2019; Izmailov et al., 2018; Achille and Soatto, 2018b).

4.2 Theoretical view and definitions of flatness.

To facilitate further discussion, we begin by looking at theoretical properties and formal definitions of sharp and flat optima.

4.2.1 Definitions of sharp and flat optima.

Hochreiter and Schmidhuber (1995) define local flatness size of the connected region around the minimum where the training loss is relatively similar. Dinh et al. (2017) formalize this as follows:

Definition 8 (ϵ - flatness)**.** *$C(\mathcal{L}, \theta, \epsilon)$ is a largest connected set containing θ, such that $\forall \theta' \in C(\mathcal{L}, \theta, \epsilon), \mathcal{L}(\theta') < \theta) + \epsilon$, where θ is a local minimum and $\mathcal{L}$ is a loss function.*

(Keskar et al., 2017) proposes to describe sharpness in terms of a maximum loss in a bounded neighbourhood of the minimum θ. (Dinh et al., 2017) formalize this as follows:

Definition 9 (ϵ - flatness)**.** *Given a Euclidean ball $B_2(\epsilon, \theta)$ centered at minimum θ with radius ϵ, the ϵ-sharpness is proportional to:*

$$\frac{\max_{\theta' \in B_2(\epsilon, \theta)} (\mathcal{L}(\theta') - \mathcal{L}(\theta))}{1 + \mathcal{L}(\theta)}, \tag{4.1}$$

where $\mathcal{L}$ is a non-negative loss function.

Finally, many authors (Chaudhari et al., 2019; Jastrzebski et al., 2018; Keskar et al., 2017) measure local curvature around the local optima with a second order

structures like Hessian and its eigenvalues. In this context, the trace of hessian or its spectral norm are used.

It was found by Sagun et al. (2017) that during training, the eigenvalues of the Hessian increase. Keskar et al. (2017) further analyzed the Hessian and related that to the flatness in loss surfaces. It was stated that flatness and the volume of the basin where we find a solution are related. On the other hand, they stated that when one moves away from the found solution, the rate of change of loss gives an estimate of flatness. Then they pointed out that these definitions are related to curvature in the loss surface, and therefore one would naturally use the Hessian of the loss function for analysis.

Keskar et al. (2017) also made a distinction between the total number of parameters and effective dimensionality. During training, there are many directions where parameters are not determined; hence the loss function is constant (equivalently, the Hessian has zero eigenvalues) along those directions. Since one does not update the parameters along those directions, they stated that those directions are degenerate. Therefore one would naturally have lower effective dimensionality compared to the number of parameters.

From the above discussion Keskar et al. (2017) observed that the function-space representation of the model is not diverse if one looks at the parameter perturbations in degenerate directions. This is related to the complexity of the model, and therefore one would expect a good generalization if the solution is in a region with lower effective dimensionality. Finally it was suggested by Keskar et al. (2017), Wu (2017) and Jastrzebski et al. (2018)that low Hessian norm region (wider minima) would most probably be reached with SGD. This is because one would reach those regions with random initialization given their larger volumes (volume of the basin of attractor) Wu (2017).

Jastrzebski et al. (2018) further analyzed the whole trajectory of SGD instead of just the final point at the end of the training. They also confirmed the analysis done by Keskar et al. (2017) and Wu (2017). A visualization of the quantitatively measures non-convexity as a heat map was given by Li et al. (2018). This was done by calculating the smallest eigenvalues of the Hessian around local minima. To facilitate further discussion, we begin by looking at theoretical properties and formal definitions of sharp and flat optima.

4.2.2 Reparametrization

One would intuitively accept the sharpness, flatness and generalization arguments. But Dinh et al. (2017) showed that one has to be careful when analyzing flatness with the Hessian of the loss function. More specifically, they gave an explicit example where a deep neural network with ReLU activation functions; one can make layer-wise weight reparameterizations that leave the network function unchanged. That would mean the same performance for the generalization. At the same time, these reparameterizations can be constructed in a way to change a sharp minimum into a wide one or a wide minimum into a sharp one. Therefore Dinh et al. (2017) stated that a sharp minimum could generalize well with an alteration to the loss function.

Later Sagun et al. (2017) noted that one has to make a non-linear transformation to deform relative widths of basins. But one can still make a meaningful analysis if one uses relative values instead of absolute ones and get a consistent comparison between various cases.

4.2.3 Bayesian interpretation of flatness

Bayesian interpretation of flatness is another intuitive and perhaps more sound way compared to Hessian. The main idea is the following, with Bayesian inference, one naturally lands into flat minima since it has a large probability mass.

Along these lines, Smith and Le (2018) argued that one can understand the sharp minima and generalization issue by evaluating the Bayesian evidence for every model. One would see that this analysis would penalize sharp minima and still be invariant under re-parameterization. Hence, this approach would resolve the concern, which was stated by Dinh et al. (2017).

One can again turn back to Keskar et al. (2017) observation: the function-space representation of the model being not diverse if one looks at the parameter perturbations in degenerate directions. Hence Occam factor would reward the flat minima but penalize sharp minima. This would lead to bad generalization for the latter case.

If one follows Smith and Le (2018) argument, that is SGD's inherent noise being the origin of its success (escaping from sharp minima), one could start thinking about modifying the learning rate. It is conceivable that one can escape from

sharp minima with a larger learning rate. Therefore there is an optimum batch size that maximizes the test accuracy for a given learning rate.

With this intuitive picture Smith and Le (2018) interpreted SGD as the discretization of a stochastic differential equation. By doing so, they predicted that the optimum batch size should scale linearly with both the learning rate and the training set size. They verified these scaling rules empirically and discussed their implications.

Dziugaite and Roy (2017) connected sharpness to Probably Approximately Correct (PAC)-Bayes bounds for the issue of good generalization. They claimed that the results of Zhang et al. (2017) could be explained by nonvacuous PAC-Bayes generalization bounds.

As Jiang et al. (2019) pointed out, as we randomly introduce noise and perturb the parameters, PAC-Bayesian interpretation captures sharpness in the expected sense. They analyzed Keskar et al. (2017) works, the part in which Keskar et al. (2017) showed a correlation between good generalization and small batch sizes. Jiang et al. (2019) named this different notion of sharpness as the worst-case sharpness. What they meant by the worst-case is the search for the direction that changes the loss the most (maximum value for the Hessian). Therefore Jiang et al. (2019) found a different PAC Bayesian measure of flatness. In their approach, insensitivity to random perturbations/flatness would give better generalization bounds compared to other bounds.

4.2.4 Flatness and simple functions

We pointed out the relation between approximately zero Hessian eigenvalues along many directions and the diversity of the function-space representation. This is closely related to Hochreiter and Schmidhuber (1995) argument: one can associate flat minima with simpler functions. Wu (2017) reiterated this point and argued that it would yield to better generalization.

The relationship between flat minima and simpler functions turned out to be a useful conceptual tool to understand the findings of Zhang et al. (2017). The behaviour of DNNs overfitting random data with almost zero training loss was studied by Arpit et al. (2017), Advani and Saxe (2017). They proposed that fitting simple functions over training data before overfitting noise is the reason behind this

good generalization behaviour.

4.2.5 Flatness and information content

Information theory can also be used to analyze the properties of flat minima. Achille and Soatto (2018a) argued that flat minima, with approximately zero eigenvalues of the Hessian along many directions, have low information content. Hence they were able to connect PAC-Bayesian approaches to information-theoretic arguments. And they further claimed that low information functions would learn invariant representations of future inputs.

The empirical findings of Entropy-SGD work of Chaudhari et al. (2019) shows a bias toward "flat minima" during optimization. After highlighting this point, Achille and Soatto (2018a) argued that these minima could be interpreted as having low information content (similar arguments appeared early on the work of Hochreiter and Schmidhuber (1997)).

Achille and Soatto (2018a) gave the following intuition: The weights of the network need not be stored with high precision since the landscape of the loss function is locally flat. Doing so would result in a smaller inference error. Therefore flat minima would generalize well, and the associated representation of the data is not sensitive to small perturbations and more disentangled.

They further argued that this approach would not have any "contradiction" with Dinh et al. (2017) result, since their argument is not "if and only if" type. That means the flatness implies low information content, but low information does not necessarily imply flatness.

4.2.6 Mode Connectivity

The usual picture that we have in our minds for a minimum of a loss function is: points at the bottom of a convex valley. As we discussed in previous sections, Keskar et al. (2017) and others argued that there is a correlation between the width of these minima and the generalization ability of the network. Li et al. (2018) also provided a low dimensional picture of the loss function of neural networks where the parameters are specified by the position of the minimum.

Immediate questions that come to mind are: "Is this picture correct? Are isolated points for global or local minima indeed lie at the bottom of some convex

valley? Are they disjoint or connected? Are we certain about the existence of bad local minima?"

Baldi and Hornik (1989) tried to address some of these questions and showed that an MLP with a single linear intermediate layer has saddle points and a global minimum but no local minima. Recently Dauphin et al. (2014) showed that "large" neural networks also have the same property. More recently, Soudry and Carmon (2016); Safran and Shamir (2016) looked at the error surface of the neural network, providing theoretical arguments for the error surface becoming well-behaved in the case of over-parametrized models.

For more modern architectures (such as CNN) Draxler et al. (2018) and Garipov et al. (2018) argued that different minima of the neural networks' loss landscape form a connected manifold, i.e. there is a path connecting local minima along which the values of the loss does not change a lot. They empirically showed that one could find a continuous path (with similar values for loss function) to connect various global minima. And Draxler et al. (2018) even conjectured that this part of the loss surface forms one single connected component.

And finally, Nguyen (2019) made a more concrete "no bad local minima" argument for neural networks. For a specific case, where one of the hidden layers has more neurons than the number of training samples, Nguyen (2019) showed that there is a continuous path (along which the value of the loss function is not increasing) from an initial point to its (asymptotic) minimal value.

We see that understanding the geometry of the loss surface is still an active research field. In relation to our this thesis, we remark that the connectivity of local optima can be useful for out of distribution tasks. That is one important motivation behind this direction of research.

4.2.7 Relation between sharpness and robustness

Several (Petzka et al., 2020; Liang et al., 2017) works tried to come up with definitions of flatness that are invariant under the reparametrization discussed above Dinh et al. (2017). Petzka et al. (2020) propose a notion of feature robustness - a representation encoded by a DNN is ϵ-robust if adding a small change in the input space X or in the features space defined by the function ϕ (feature encoder) does not change the empirical error by more then ϵ. Formally, Petzka et al. (2020) define

this as follows:

Definition 10 ((δ, S, ϵ)- feature robustness)**.** *Given:*

$$\mathcal{F}(\delta, S, A) := \frac{1}{|S|} \sum_{(x,y)\in S} [l(\psi(\phi(x) + \delta A\phi(x)), y) - l(\{(x), y)], \tag{4.2}$$

where $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ *denote a loss function,* δ *and* ϵ *are two small positive numbers,* $S = \{(x_i, y_i)_{i=1}^N\} \subseteq \mathcal{X} \times \mathcal{Y}$ *is the training set, and* $A \in \mathbb{R}^{m\times m}$ *is a feature selection matrix such that* $||A|| \leq 1$*. A model* $f(x) = (\psi \circ \phi)(x)$*, where* $\phi : \mathcal{X} \to \mathbb{R}^m$ *is a feature extractor and* $\psi : \mathbb{R}^m \to \mathcal{Y}$ *is a mapping to the output space (e.g. classifier), is called* $((\delta, S, A), \epsilon)$ *- feature robust, if* $|\mathcal{F}(\delta', S, A)| < \epsilon$*,* $\forall|\delta'| \leq \delta$*.*

Furthermore, Petzka et al. (2020) show that feature robustness of a model $f(\mathbf{w}, x) = \psi(\mathbf{w}, \phi(x)) = g(\mathbf{w}\phi(x)$, ($g$ is a twice differentiable function) can be bounded as follows:

$$\max_{||A||\leq 1} \mathcal{F}(\delta, S, A) \leq \frac{\delta^2}{2}||w_l||^2 \cdot \lambda_{max}^{H,l}(w_l) + \mathcal{O}(\delta^3), \tag{4.3}$$

where $\lambda_{max}^H(w_*)$ is the largest eigenvalue of the Hessian $H_{\mathcal{E}_{emp}}(w_*, S)$ of the empirical error at w_*, with $w*$ being a minimizer of the empirical error $\mathcal{E}_{emp}$.

Petzka et al. (2020) further propose a measure $\kappa^\phi(w)$ of flatness of loss landscape (lower values indicate more flatness). For a neural network with L layers, a measure of flatness for each layer ℓ is defined as:

$$\kappa^\phi(w) := ||w_\ell||^2 \cdot \lambda_{max}^{H,\ell}(w_\ell). \tag{4.4}$$

They show that this definition of flatness is invariant under the reparameterization and relates to feature robustness as follows.

Definition 11 ((δ, S, ϵ) - feature robustness of a neural network)**.** *A neural network with* L *layers is* $((\delta, S, A), \epsilon)$ *- feature robust in layer* ℓ*,* $1 \leq \ell \leq L$*, at* w_* *for* $\epsilon = \frac{\delta^2}{2}\kappa^\ell(w_*) + \mathcal{O}(\delta^3)$*.*

Under the assumption of representativeness of the training set S, authors further derive a bound on of the generalization error.

Achille and Soatto (2018a) rely on the information theoretical view of feature robustness. More specifically, they refer to the Information Bottleneck Theory (Tishby and Zaslavsky, 2015), which defines the optimal representation $z = f(x)$ of the high dimensional raw data x and associated label y to be (a) sufficient for the task (i.e. $I(y; z) = I(y; x)$, with I denoting mutual information) and (b) minimal (i.e. minimal $I(z; x)$, which leads to minimal amount of information stored in the weights). Achille and Soatto (2018a) further propose (c) invariance ($I(z; n) = 0$, where n is a nuisance such as e.g. translation, rotation, occlusion, s.t. $yn|x$) and (d) disentanglement as additional properties of an optimal representation. They further show that due to the noisy nature of the optimization algorithms (e.g. SGD), in practice, it only suffices to ensure (a) and (b) through implicit or explicit regularization during the training process.

The notion of an optimal representation has received a lot of attention in the machine learning research community lately, mostly due to the fact that self-supervised representation learning methods (Chen et al., 2020; He et al., 2019) have reached the state-of-the-art performance dictated by supervised learning algorithms.

4.3 Algorithms

One of the main motivations behind most of the works that we covered is: understanding the "unexpectedly" good generalization behaviour of SGD (even in situations when the number of parameters of the model largely exceeds the number of training data points). If one can find the underlying source of SGD's success, then one could focus on those properties and use them to come up with algorithms that are more efficient than SGD.

Chaudhari et al. (2019) proposed the Entropy-SGD algorithm with the assumption that flat/wide optima is the main source of SGD's success. They used Gibbs distribution to model local entropy and minimized that modified loss function $F(x, \gamma)$ which is defined in equation A.1.[1] Their Entropy-SGD algorithm had an "SGD inside an SGD" form in two nested loops. To calculate the gradients, they

1. In their work Chaudhari et al. (2019) claimed to prove that the modified objective function is smoother than SGD and generalizes better than SGD. In the Appendix chapter, we went over their proof and showed where their proof is incorrect.

used Langevin dynamics in the inner loop before the weight updating step. With that specific form of the modified loss function that they used, they were able to get a smoother landscape and reach the flat minima faster than SGD while avoiding poorly-generalizable solutions located in the sharp valleys.

Achille and Soatto (2018b) chose a different approach and focused on the noisiness of SGD to be the source of its success. They proposed a technique called Information Dropout, which can be seen as a generalization of dropout, where their proposed regularizer reduces to binary dropout Srivastava et al. (2014) when using Bernoulli type of noise. Importantly, the amount of noise added to the activations of a layer is **dependent** on the activation of the layer itself, thus penalizing a large amount of information passed between the layers leading to minimal and sufficient representation in the spirit of the Information Bottleneck theory Tishby and Zaslavsky (2015). In a later work Achille and Soatto (2018a) showed that the quality of the stochastic representation depends on the flatness of the minimum. With that in mind, they focused on the learning of optimal representations to get the most out of the data for a specific task.

As we discussed before, one can also argue that it is not only the batch size of the SGD algorithm but rather the ratio of the learning rate and the batch size being the main source of the success of SGD. Izmailov et al. (2018) showed that simple averaging of multiple points along the trajectory of SGD, with a cyclical or constant learning rate, leads to better generalization than conventional training. They also showed that this Stochastic Weight Averaging (SWA) procedure finds much flatter solutions than SGD.

4.4 Flatness Conjecture and Experiments

In machine learning, great successes have been achieved in language modelling (Brown et al., 2020) and image classification (Dosovitskiy et al., 2020) at levels that exceed human-level performance. But when the test data distribution is slightly different than the training domain, then we see drastic changes in performance (Su et al., 2019). Understanding the reasons behind failing to do so is still an open problem.

It is not very clear if the EL-ID task might be considered as an OOD problem. It is true that changing one pixel, causing a degradation of the performance, happens when the circumstances are carefully designed (Su et al., 2019). But Recht et al. (2019) showed that when new images are used that are not from the excessively re-used test sets, performance drops more than 10%. Therefore we believe that methods to improve OOD generalization are quite relevant to our EL-ID task, no matter how similar our MC produced data and real data are.

There are recent efforts to understand the reason behind the failure of OOD generalization. Arjovsky et al. (2019) claimed that it is related to the models failing to capture the causal factors of variation in data and instead learn spurious correlations that change from training set to data set. They suggested a method, Invariant Risk Minimization (IRM), that would learn features that are not spurious. But very recently Gulrajani and Lopez-Paz (2020) showed that none of the modern algorithms, including IRM, outperform ERM by a significant margin when all conditions are equal.

In this work we make the following conjecture:

Flat optimum has an advantage over the sharp one in their out of distribution generalization performances. To test our conjecture, we will use two algorithms for our EL-ID data set.

4.4.1 Entropy-SGD and SWA Algorithms

We implemented both Entropy-SGD and SWA methods to the high granularity 56x11 pixel sized calorimeter images. Since, at this moment, we did not have access to real calorimeter images, we divided the simulated data into two parts based on their angular position values η. Dividing the calorimeter images into $|\eta| < 0.65$ and $|\eta| \geq 0.65$ will serve as a surrogate for real data vs simulated data distinction.

As we can see from Figure 4.1 and Figure 4.2 algorithms[2] that drift the parameters towards with wide flat minimums of training loss surface indicated more stable learning curves. For all of these cases, we used the same architecture with

2. We should note that the Figure 4.1 and Figure 4.2 are of SWA algorithm since it is the only one that we can compare epoch per epoch with SGD. Because 1 epoch for an Entropy-SGD training corresponds to L epochs of SGD training, where L is the Langevin iterations parameter. The general shape of the learning curve of SWA is very similar to Entropy-SGD, only scaled by a factor of L along the x-axis. Since it would be confusing to compare Entropy-SGD vs. we chose to put SWA vs. SGD results.

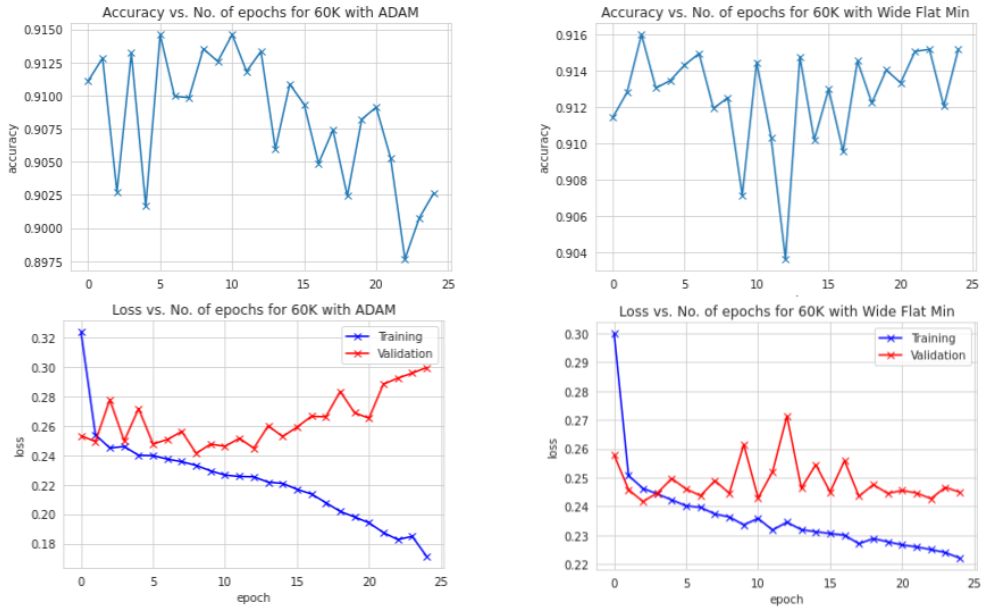

Figure 4.1 – Learning curve comparison of algorithms that chooses wide flat optimums of the loss function with ADAM for 60K events. Training set is chosen to be $|\eta| < 0.65$, and validation set $|\eta| \geq 0.65$.

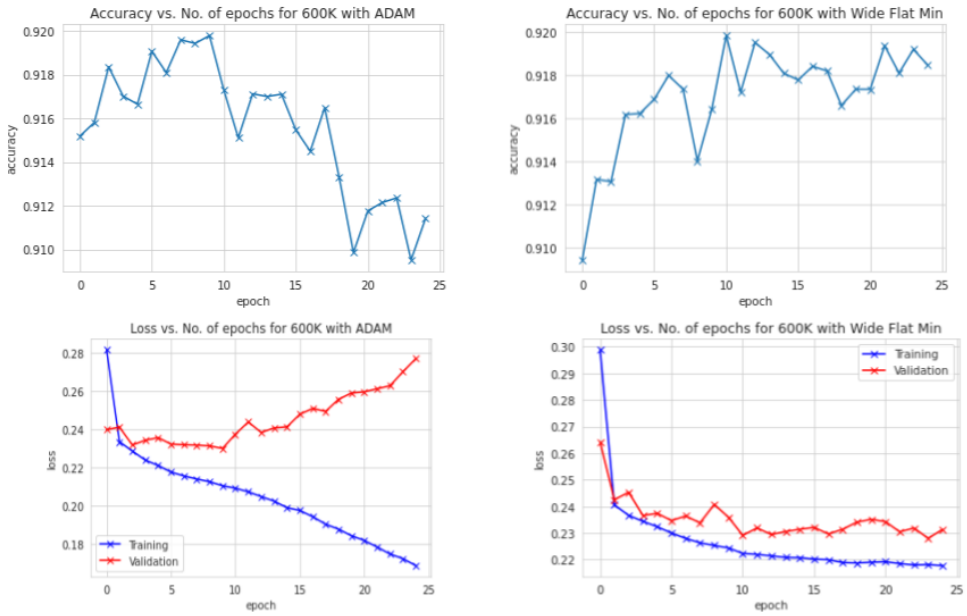

Figure 4.2 – Learning curve comparison of algorithms that chooses wide flat optimums of the loss function with ADAM for 600K events. Training set is chosen to be $|\eta| < 0.65$, and validation set $|\eta| \geq 0.65$.

the same number of parameters. The two algorithms gave very similar results and consistently performed better than ADAM. We also noticed that as soon as we increased the number of events from 60K to 600K [3] then, the oscillatory behaviour of the validation loss curve of ADAM started to get smoother. But in both of these cases, algorithms that lead the model towards its wide flat minimum performed better, achieved higher accuracy and less over-fitting [4].

However, we can not decisively claim that our conjecture is verified since we did not test our model, which was trained with simulated data, with the real observed data obtained by the ATLAS detector. The distinction was a surrogate of the real out of distribution situation that would arise when we test our model with the real labelled data.

Finally, we trained the full architecture with 14M events and used all available data (scalars, tracks, calorimeter images). In that case the change in best accuracy was less than 0.2% when the model was trained in $|\eta| < 0.65$ and tested in $|\eta| \geq 0.65$ region or $|\eta| < 0.65$ region (IID). Therefore we can see that when we increase the number of events and include other parts of the data, OOD surrogate behaviour gets lost.

We attribute this to two factors. Firstly if two domains (train and test) are not entirely different, more training with more data will help with OOD generalization performance. But if the two domains in question are very different, then training with more data can make our model overfit the training set and generalize even less to the test set.

The second factor is the weaker dependency of scalar and track parts of our data to the value of angular position $|\eta|$, compared to calorimeter images. Even when there is, the possible asymmetry of the dependency made the effect smaller.

Our conjecture on the relation between OOD generalization performance and flat wide minimum still holds. It would be interesting to see the future results for different datasets and algorithms that might lessen the problems associated with domain adaptation.

3. We should note that these numbers are the sizes of the training set events. Throughout the whole work, we used 10% for the test set, but here it had to be 50% since the data is evenly distributed between $|\eta| < 0.65$ and $|\eta| \geq 0.65$. In short, the number of total events here is 1.2M, where half of them are used for training and the other half for testing.

4. Training with high η and testing with low η gave similar results. The change in best accuracy was 0.6% for 60K events and 0.4% for 600K events, when training and validating are chosen in the same η range(IID)

5 Conclusion

This thesis solves the electron identification task by using tools of machine learning. This is a very important task since electrons are essential to study physics at the LHC. They have already played an important role in the discovery of the Higgs boson, which was awarded a Nobel prize. In the future, they might play an important role in other discoveries such as supersymmetry or the discovery of dark matter at the LHC. Our main objective was to outperform the currently used LH algorithm in identifying electrons in the ATLAS detector with machine learning algorithms.

In this thesis, we show that our model reaches a performance that is 18 times better than the standard LH algorithms that are currently used by the ATLAS collaboration. An important factor in achieving high accuracy with a learning model is choosing the right architecture. This choice depends on the structure of the dataset at hand. By using CNNs combined with FCNs, we are able to make use of the sequential nature of our data.

During our experiments, we observed that the two most important factors to increase accuracy are training with more data and using all parts of the data. When we use different parts of our data (images, scalars and tracks) separately, the accuracy is not high. But when used together, our model is able to reach its best performance.

At the end of our experiments, we also conclude that a deeper or wider model does not lead to an increase in accuracy for our dataset. For the EL-ID task going very deep or very wide only led to a longer time for training but no improvement in the accuracy.

Although we are able to get high accuracy with our model for simulated data, we would like to know how our algorithm does for identifying real electrons that will be collected by ATLAS. Training a model with simulated data and applying it to real data is an OOD task. Therefore we aim to get a good OOD generalization performance. In our work, we investigated the answer to a specific question: What is

the effect of wide flat minima on OOD generalization performance? We conjecture that it is possible to get better generalization performance with Entropy-SGD and SWA like methods that favour solutions that lie in wide flat minima of the training loss surface for out of distribution tasks. We implemented these algorithms to our dataset and got results that supported our conjecture.

There are other methods that can help with both the OOD performance of our model when we test it with real data. One immediate future work would be using the Replicated Stochastic Gradient Descent (rSGD) method (Baldassi et al., 2016) instead of Entropy-SGD or SWA. It would be interesting to see the OOD generalization performance of the rSGD method on our EL-ID dataset.

Data augmentation can also help with the OOD generalization performance of a model. CutOut is a method that is similar to dropout, which we discussed in previous chapters, where one drops certain random parts of an image input before feeding it to the model. It was shown that the CutOut method improves the generalization performance of various models (Devries and Taylor, 2017).

Mix-up is another data augmentation method that makes linear interpolation between images and output labels (Zhang et al., 2018). It also improved the generalization of state-of-the-art models at the time.

Recently the CutMix method, which is a combination of the CutOut and Mix-up methods, is also shown to outperform the generalization performance of the state-of-the-art augmentation strategies on CIFAR and ImageNet.

More recently, Manifold Mixup was proposed, which is similar to the Mix-up method. But it not only does a linear interpolation at the level of input images and labels but also at the intermediate layers of the neural network. (Verma et al., 2019) showed that they achieved not only state-of-the-art results but also made the models robust to adversarial attacks.

To conclude, we believe that the model presented in this thesis will undoubtedly help the ATLAS collaboration do much better in the electron identification task. Hence we are hopeful that our algorithm will take the place of the current method that ATLAS collaboration uses for solving this important task.

A Theorems, Lemmas and Proofs

A.0.1 Smoothness Proof

We would like to point out that Lemma 1 of "Entropy-SGD: biasing gradient descent into wide valleys" (Chaudhari et al., 2019) is incorrect. Theorem 13, which is the main theorem of the same paper, and it is based on the correctness of Lemma 1, is also incorrect. Therefore one can not (with the way it is stated in the paper) prove that Entropy-SGD has a smaller generalization error than the original objective by using the main theorem in (Chaudhari et al., 2019). Let us start by directly quoting the relevant parts of their paper as well as the problematic proof that they give.

Definition 12 (**Local entropy**).

$$F(x, \gamma) = \log \int_{x'} \exp\Big(- f(x') - \frac{\gamma}{2} \, \|x - x'\|_2^2 \Big) \, dx'. \tag{A.1}$$

Here $f(x)$ is the original loss function, and γ is a hyper-parameter that is used to seek out valleys of specific widths on the energy landscape.

Lemma 1. *The objective $F(x, \gamma; \Xi)$ in (6) is $\frac{\alpha}{1+\gamma^{-1} \, c}$-Lipschitz and $\frac{\beta}{1+\gamma^{-1} \, c}$-smooth.*

The local entropy objective is thus smoother than the original objective.

Theorem 13. *For an α-Lipschitz and β-smooth loss function, if SGD converges in T iterations on N samples with decreasing learning rate $\eta_t \leq 1/t$ the stability is bounded by*

$$\epsilon \lessapprox \frac{1}{N} \, \alpha^{1/(1+\beta)} \, T^{1-1/(1+\beta)}.$$

Using Lemma 1 and Theorem 13 we have

$$\epsilon_{\text{ Entropy-SGD}} \lessapprox \Big(\alpha \, T^{-1} \Big)^{\left(1 - \frac{1}{1+\gamma^{-1} c}\right) \, \beta} \, \epsilon_{\text{ SGD}}, \tag{A.2}$$

which shows that Entropy-SGD generalizes better than SGD for all $T > \alpha$ if they both converge after T passes over the samples.

Proof of Lemma 1. The gradient $-\nabla F(x)$ is computed to be $\gamma \Big(x - \langle x';\ \Xi^{\ell} \rangle\Big)$. Consider the term

$$\begin{aligned}
x - \langle x';\ x \rangle &= x - Z_{x,\gamma}^{-1} \int_{x'} x'\ e^{-f(x') - \frac{\gamma}{2}\ \|x - x'\|^2}\ dx' \\
&\approx x - Z_{x,\gamma}^{-1} \int_{s} (x+s)\ e^{-f(x) - \nabla f(x)^{\top} s - \frac{1}{2}\ s^{\top} \Big(\gamma + \nabla^2 f(x)\Big) s}\ ds \\
&= x \Big(1 - Z_{x,\gamma}^{-1} \int_{s} e^{-f(x) - \nabla f(x)^{\top} s - \frac{1}{2}\ s^{\top} \Big(\gamma + \nabla^2 f(x)\Big) s}\ ds\Big) \\
&- Z_{x,\gamma}^{-1} \int_{s} s\ e^{-f(x) - \nabla f(x)^{\top} s - \frac{1}{2}\ s^{\top} \Big(\gamma + \nabla^2 f(x)\Big) s}\ ds \\
&= -Z_{x,\gamma}^{-1}\ e^{-f(x)} \int_{s} s\ e^{-\nabla f(x)^{\top} s - \frac{1}{2}\ s^{\top} \Big(\gamma + \nabla^2 f(x)\Big) s}\ ds.
\end{aligned}$$

The above expression is the mean of a distribution $\propto e^{-\nabla f(x)^{\top} s - \frac{1}{2}\ s^{\top} \Big(\gamma + \nabla^2 f(x)\Big) s}$. We can approximate it using the saddle point method as the value of s that minimizes the exponent to get

$$x - \langle x';\ x \rangle \approx \Big(\nabla^2 f(x) + \gamma\ I\Big)^{-1}\ \nabla f(x).$$

Let us denote $A(x) := \Big(I + \gamma^{-1}\ \nabla^2 f(x)\Big)^{-1}$. Plugging this into the condition for smoothness, we have

$$\begin{aligned}
\|\nabla F(x, \gamma) - \nabla F(y, \gamma)\| &= \|A(x)\ \nabla f(x) - A(y)\ \nabla f(y)\| \\
&\leq \Big(\sup_{x}\ \|A(x)\|\Big)\ \beta\ \|x - y\|.
\end{aligned}$$

Unfortunately, we can only get a uniform bound if we assume that for a small constant $c > 0$, no eigenvalue of $\nabla^2 f(x)$ lies in the set $[-2\gamma - c, c]$. This gives

$$\Big(\sup_{x}\ \|A(x)\|\Big) \leq \frac{1}{1 + \gamma^{-1}\ c}.$$

This shows that a smaller value of γ results in a smoother energy landscape, except

at places with very flat directions. The Lipschitz constant also decreases by the same factor. □

After having quoted the relevant parts of the Chaudhari et al. (2019) paper, let us look at the problematic step:

$$\|A(x)\ \nabla f(x) - A(y)\ \nabla f(y)\| \leq \Big(\sup_x\ \|A(x)\|\Big)\ \beta\ \|x - y\|.$$

We will first point out the key point where the above inequality fails to hold. And then, we will give a counter-example to show that the above equation is not valid for all $f(x)$ and $A(x)$ functions.

First we will add and subtract the same term to the left hand side of the above equation

$$\begin{aligned}
&\|A(x)\ \nabla f(x) - A(y)\ \nabla f(y)\| \\
&\qquad = \|A(x)\ \nabla f(x) - A(y)\ \nabla f(x) + A(y)\ \nabla f(x) - A(y)\ \nabla f(y)\| \\
&\qquad \leq \|A(x) - A(y)\|\|\nabla f(x)\| + \|A(y)\|\|\nabla f(x) - \nabla f(y)\| \\
&\qquad \leq \|A(x) - A(y)\|\|\nabla f(x)\| + \Big(\sup_x\ \|A(x)\|\Big)\ \beta\ \|x - y\|.
\end{aligned}$$

One can see that given the extra term that appears above, it is impossible conclude that the new objective function $F(x, \gamma;\ \Xi)$ is $\frac{\beta}{1+\gamma^{-1}\ c}$-smooth.

Now we would like to give an explicit counter-example to show that Lemma 1 does not hold for all functions with the stated properties in the paper. To make things simpler let us assume that both $f(x)$ and $A(x)$ are $\mathcal{R} \to \mathcal{R}$. Hence ∇ will become an ordinary derivative and we will further rename $\nabla f(x) = g(x)$. Therefore the claim of Lemma 1 is the following

$$\|A(x)\ g(x) - A(y)\ g(y)\| \leq \Big(\sup_x\ \|A(x)\|\Big)\ \beta\ \|x - y\|.$$

First we will take a β smooth function $f(x)$ to have its derivative to be symmetric under inversion.

$$\text{For}\ \ x = a\ \ \text{and}\ \ y = -a \Rightarrow g(a) = g(-a).$$

The inequality in question becomes

$$\begin{aligned}\|A(a)\ g(a) - A(-a)\ g(-a)\| &= \|A(a) - A(-a)\|\|g(a)\| \\ &\overset{?}{\leq} \Big(\sup_x\ \|A(x)\|\Big)\ \|g(a) - g(-a)\| = 0.\end{aligned}$$

The only way that the above equation is correct is if $A(x)$ is also symmetric under inversion so that $\|A(a) - A(-a)\| = 0$. But $A(x)$ is certainly not invariant under inversion for all functions $f(x)$ in general.

Let us disprove the above claim by giving an explicit counter-example

$$f(x) = \frac{1}{3}x^3 \ \text{ and } \ A(x) = \Big(I + \gamma^{-1}\ \nabla^2 f(x)\Big)^{-1} = \frac{\gamma}{\gamma + 2x}.$$

Notice that for $x \in (-\frac{\gamma}{2}, \frac{\gamma}{2})$ $f(x)$ is γ-smooth and $A(x)$ is finite and fulfills the required conditions. This would give $g(x) = x^2$ that is symmetric under inversion; $g(x) = g(-x)$. On the other hand, we can see that $A(x)$ is not invariant under inversion $A(a) \neq A(-a)$.

To sum up the proof that Chaudhari et al. (2019) gave for Lemma 1 is incorrect. That would mean that the proof of Theorem 3 of their paper is also incorrect. That would mean that the **proof** of "Entropy-SGD generalizes better than SGD" claim is incorrect. But we do not conclude that their **claim** of "Entropy-SGD generalizes better than SGD" claim is incorrect. [1]

1. After pointing this error to the authors, they admitted their error in proving Lemma 2, consequently the main theorem (Theorem 3) in their paper. The authors pointed us to a follow-up paper Chaudhari et al. (2017), where they proved the smoothness of local entropy without even making any Laplace approximation. The smoothness proof can be seen from Lemma 18 of Chaudhari et al. (2017) where they used the maximum principle in partial differential equation theory and the fact that the trace of the Hessian is an upper bound on the smoothness parameter for doubly-differentiable functions.

Bibliography

Achille, A. and S. Soatto (2018a). Emergence of invariance and disentanglement in deep representations. *The Journal of Machine Learning Research 19*(1), 1947–1980.

Achille, A. and S. Soatto (2018b). Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence 40*(12), 2897–2905.

Advani, M. S. and A. M. Saxe (2017). High-dimensional dynamics of generalization error in neural networks. *ArXiv abs/1710.03667*.

Arjovsky, M., L. Bottou, I. Gulrajani, and D. Lopez-Paz (2019). Invariant risk minimization. *ArXiv abs/1907.02893*.

Arpit, D., S. Jastrzebski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. C. Courville, Y. Bengio, and S. Lacoste-Julien (2017). A closer look at memorization in deep networks. In *ICML*.

ATLAS Collaboration (2012). Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Physics Letters B 716*, 1–29.

ATLAS Collaboration (2014). Measurement of the Z/γ^*boson transverse momentum distribution in pp collisions at $\sqrt{s} = 7$ TeV with the ATLAS detector. *Journal of High Energy Physics 2014*, 1–47.

ATLAS Collaboration (2019). Electron reconstruction and identification in the ATLAS experiment using the 2015 and 2016 LHC proton-proton collision data at $\sqrt{s} = 13$ TeV. *European Physical Journal C 79*, 639.

Bahdanau, D., K. Cho, and Y. Bengio (2015). Neural machine translation by jointly learning to align and translate. *CoRR abs/1409.0473*.

Baldassi, C., C. Borgs, J. Chayes, A. Ingrosso, C. Lucibello, L. Saglietti, and R. Zecchina (2016). Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes. *Proceedings of the National Academy of Sciences 113*, E7655 – E7662.

Baldi, P. and K. Hornik (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks 2*, 53–58.

Baldi, P., P. Sadowski, and D. Whiteson (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature communications 5*, 4308.

Belkin, M., D. Hsu, S. Ma, and S. Mandal (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences 116*, 15849 – 15854.

Blei, D., A. Ng, and M. I. Jordan (2003). Latent dirichlet allocation. *J. Mach. Learn. Res. 3*, 993–1022.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krüger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei (2020). Language models are few-shot learners. *ArXiv abs/2005.14165*.

Carreira, J. and A. Zisserman (2017). Quo vadis, action recognition? a new model and the kinetics dataset. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4724–4733.

Chaudhari, P., A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina (2019). Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment 2019*(12), 124018.

Chaudhari, P., A. M. Oberman, S. Osher, S. Soatto, and G. Carlier (2017). Deep relaxation: partial differential equations for optimizing deep neural networks. *Research in the Mathematical Sciences 5*, 1–30.

Chen, T., S. Kornblith, M. Norouzi, and G. Hinton (2020). A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*.

CMS Collaboration (2012). Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Physics Letters B 716*, 30–61.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems 2*(4), 303–314.

Dauphin, Y., R. Pascanu, Çaglar Gülçehre, K. Cho, S. Ganguli, and Y. Bengio (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS*.

Devries, T. and G. W. Taylor (2017). Improved regularization of convolutional neural networks with cutout. *ArXiv abs/1708.04552*.

Diggle, P. and R. Gratton (1984). Monte carlo methods of inference for implicit statistical models. *Journal of the royal statistical society series b-methodological 46*, 193–212.

Dinh, L., R. Pascanu, S. Bengio, and Y. Bengio (2017). Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1019–1028. JMLR. org.

Donahue, J., L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell (2015). Long-term recurrent convolutional networks for visual recognition and description. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2625–2634.

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv abs/2010.11929*.

Draxler, F., K. Veschgini, M. Salmhofer, and F. A. Hamprecht (2018). Essentially no barriers in neural network energy landscape. *ArXiv abs/1803.00885*.

Dziugaite, G. K. and D. M. Roy (2017). Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*.

Feichtenhofer, C., A. Pinz, and A. Zisserman (2016). Convolutional two-stream network fusion for video action recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1933–1941.

Garipov, T., P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson (2018). Loss surfaces, mode connectivity, and fast ensembling of dnns. In *NeurIPS*.

Girdhar, R., J. Carreira, C. Doersch, and A. Zisserman (2019). Video action transformer network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 244–253.

Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014). Generative adversarial networks. In *NIPS'2014*.

Gulrajani, I. and D. Lopez-Paz (2020). In search of lost domain generalization. *ArXiv abs/2007.01434*.

He, K., H. Fan, Y. Wu, S. Xie, and R. Girshick (2019). Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*.

He, K., X. Zhang, S. Ren, and J. Sun (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–1034.

He, K., X. Zhang, S. Ren, and J. Sun (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Henighan, T., J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. Brown, P. Dhariwal, S. Gray, C. Hallacy, B. Mann, A. Radford, A. Ramesh, N. Ryder, D. Ziegler, J. Schulman, D. Amodei, and S. McCandlish (2020). Scaling laws for autoregressive generative modeling. *ArXiv abs/2010.14701*.

Hinton, G. and D. von Cramp. Keeping neural networks simple by minimising the description length of weights. 1993. In *Proceedings of COLT-93*, pp. 5–13.

Hochreiter, S. and J. Schmidhuber (1995). Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*, pp. 529–536.

Hoecker, A., P. Speckmayer, J. Stelzer, J. Therhaag, E. V. Toerne, H. Voss, M. Backes, T. Carli, O. Cohen, A. Christov, D. Dannheim, K. Danielowski, S. Henrot-Versill'e, M. Jachowski, K. Kraszewski, A. Krasznahorkay, M. Kruk, Y. Mahalalel, R. Ospanov, X. Prudent, A. Robert, D. Schouten, F. Tegenfeldt, A. Voigt, K. Voss, M. Wolter, and A. Zemla (2007). Tmva - toolkit for multivariate data analysis.

Huang, G., Z. Liu, and K. Q. Weinberger (2017). Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269.

Ioffe, S. and C. Szegedy (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Izmailov, P., D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson (2018). Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.

Jastrzebski, S., Z. Kenton, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey (2018). On the relation between the sharpest directions of dnn loss and the sgd step length. *arXiv preprint arXiv:1807.05031*.

Jiang, Y., B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio (2019). Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*.

Karpathy, A., G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei (2014). Large-scale video classification with convolutional neural networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1725–1732.

Keskar, N., D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. Tang (2017). On large-batch training for deep learning: Generalization gap and sharp minima. *ArXiv abs/1609.04836*.

Kingma, D. P. and M. Welling (2014). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Li, H., Z. Xu, G. Taylor, and T. Goldstein (2018). Visualizing the loss landscape of neural nets. In *NeurIPS*.

Liang, T., T. Poggio, A. Rakhlin, and J. Stokes (2017). Fisher-rao metric, geometry, and complexity of neural networks. *arXiv preprint arXiv:1711.01530*.

Liu, R., J. Lehman, P. Molino, F. P. Such, E. Frank, A. Sergeev, and J. Yosinski (2018). An intriguing failing of convolutional neural networks and the coordconv solution. *ArXiv abs/1807.03247*.

Lu, Z., H. Pu, F. Wang, Z. Hu, and L. Wang (2017). The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems*, pp. 6231–6239.

Metz, L., N. Maheswaranathan, C. D. Freeman, B. Poole, and J. Sohl-Dickstein (2020). Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves. *ArXiv abs/2009.11243*.

Mochizuki, K. (2020). *TF framework for the identification of ATLAS electrons by using neural networks*. https://github.com/mociduki/el_classifier.

Neal, R. (1992). Connectionist learning of belief networks. *Artif. Intell. 56*, 71–113.

Neyshabur, B., R. Tomioka, and N. Srebro (2015). In search of the real inductive bias: On the role of implicit regularization in deep learning. *CoRR abs/1412.6614*.

Nguyen, Q. (2019). On connected sublevel sets in deep learning. *ArXiv abs/1901.07417*.

Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of Computation 35*, 773–782.

Petzka, H., L. Adilova, M. Kamp, and C. Sminchisescu (2020). Feature-robustness, flatness and generalization error for deep neural networks. *arXiv preprint arXiv:2001.00939*.

Razin, N. and N. Cohen (2020). Implicit regularization in deep learning may not be explainable by norms. *ArXiv abs/2005.06398*.

Recht, B., R. Roelofs, L. Schmidt, and V. Shankar (2019). Do imagenet classifiers generalize to imagenet? In *ICML*.

Safran, I. and O. Shamir (2016). On the quality of the initial basin in overspecified neural networks. *ArXiv abs/1511.04210*.

Sagun, L., U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou (2017). Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454* .

Schmidt, R. M., F. Schneider, and P. Hennig (2020). Descending through a crowded valley - benchmarking deep learning optimizers. *ArXiv abs/2007.01547*.

Shi, X., Z. Chen, H. Wang, D. Yeung, W. Wong, and W. chun Woo (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *ArXiv abs/1506.04214*.

Simonyan, K. and A. Zisserman (2014). Two-stream convolutional networks for action recognition in videos. In *NIPS*.

Smith, S. L., P. Kindermans, and Q. V. Le (2018). Don't decay the learning rate, increase the batch size. *ArXiv abs/1711.00489*.

Smith, S. L. and Q. V. Le (2018). A bayesian perspective on generalization and stochastic gradient descent. *ArXiv abs/1710.06451*.

Soudry, D. and Y. Carmon (2016). No bad local minima: Data independent training error guarantees for multilayer neural networks. *ArXiv abs/1605.08361*.

Srivastava, N., G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res. 15*, 1929–1958.

Srivastava, N., E. Mansimov, and R. Salakhutdinov (2015). Unsupervised learning of video representations using lstms. *ArXiv abs/1502.04681*.

Su, J., D. V. Vargas, and K. Sakurai (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation 23*, 828–841.

Tishby, N. and N. Zaslavsky (2015). Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE.

Vapnik, V. (1971). Chervonenkis: On the uniform convergence of relative frequencies of events to their probabilities.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention is all you need. *ArXiv abs/1706.03762*.

Verma, V., A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio (2019). Manifold mixup: Better representations by interpolating hidden states. In *ICML*.

Wu, L. (2017). Towards understanding generalization of deep learning : Perspective of loss landscapes.

Yang, G., J. Pennington, V. Rao, J. Sohl-Dickstein, and S. Schoenholz (2019). A mean field theory of batch normalization. *ArXiv abs/1902.08129*.

Yao, L., A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. C. Courville (2015). Describing videos by exploiting temporal structure. *2015 IEEE International Conference on Computer Vision (ICCV)*, 4507–4515.

Zagoruyko, S. and N. Komodakis (2016). Wide residual networks. *ArXiv abs/1605.07146*.

Zhang, C., S. Bengio, M. Hardt, B. Recht, and O. Vinyals (2017). Understanding deep learning requires rethinking generalization. *ArXiv abs/1611.03530*.

Zhang, H., M. Cissé, Y. Dauphin, and D. Lopez-Paz (2018). mixup: Beyond empirical risk minimization. *ArXiv abs/1710.09412*.